Question part

Rest api

- ## Basic & Conceptual Questions

What are the different ways to build APIs in Drupal 10?

What is the difference between REST, JSON:API, and GraphQL in Drupal?

Which core modules are required to enable REST in Drupal 10?

How do you enable REST for a specific entity type (e.g., node)?

What is serialization in Drupal? Which modules handle it?

What is the role of the Serializer and Normalizer in Drupal REST?

- ## Configuration & Setup

How do you configure REST resources in Drupal 10?

How do you enable authentication methods for REST (Basic Auth, Cookie, OAuth)?

What is the difference between Basic Authentication and Cookie-based Authentication in Drupal?

How do you test REST APIs in Drupal (tools and process)?

How do you expose custom fields in REST responses?

How do you restrict access to REST endpoints based on roles?

- **JSON:API Specific (Very Important in Drupal 10)**

Why is JSON:API preferred in Drupal 10?

How do you filter content using JSON:API?

How do you include related entities in a

JSON:API response?

How do you sort and paginate results in JSON:API?

How do you disable a specific resource in JSON:API?

- **Custom REST Resource Development**

How do you create a custom REST resource in Drupal 10?

What annotation is used to define a REST resource plugin?

Which class do you extend while creating a custom REST resource?

How do you define GET, POST, PATCH, DELETE methods in a custom REST resource?

How do you handle validation inside a custom REST endpoint?

How do you return custom response codes

(200, 201, 403, 404)?

- Security & Performance

How do you secure REST APIs in Drupal?

How do you implement CSRF protection?

How does caching work in Drupal REST?

How do cache contexts and cache tags affect API responses?

How do you disable caching for a specific REST response?

- Practical / Scenario-Based

A mobile app needs to create content in Drupal. How would you implement it?

How would you version your API in Drupal?

How do you handle file/image uploads via REST?

How do you debug REST API issues in

Drupal?

What common REST errors have you faced and how did you fix them?

Answer part

- **Basic & Conceptual Questions**

Drupal 10 supports REST using core REST module, JSON:API module, and contributed GraphQL module.

REST is configurable and flexible, JSON:API is standards-compliant and auto-exposes entities, and GraphQL allows client-driven queries.

The required core modules are REST, Serialization, and HAL (if using HAL format).

REST is enabled per entity and method under /admin/config/services/rest.

Serialization converts Drupal objects into formats like JSON or XML using the Serialization module.

Serializer converts data formats while Normalizers transform Drupal objects into structured arrays for serialization.

- **Configuration & Setup**

REST resources are configured from the REST UI where methods, formats, and authentication providers are enabled.

Authentication methods are enabled by activating modules like Basic Auth, OAuth, or using Cookie authentication for logged-in users.

Basic Auth uses username and password per request while Cookie authentication works for authenticated session users.

REST APIs are tested using tools like

Postman or cURL by sending HTTP requests to endpoints.

Custom fields are exposed automatically if attached to the entity and proper permissions are granted.

Access is restricted using Drupal permissions assigned to specific roles.

- **JSON:API Specific**

JSON:API is preferred because it is included in core and requires zero configuration to expose entities.

Filtering in JSON:API is done using query parameters like ?filter[field_name]=value.

Related entities are included using the include query parameter.

Sorting and pagination are handled using sort and page[limit] query parameters.

Specific resources can be disabled using the JSON:API Extras module.

- **Custom REST Resource Development**

A custom REST resource is created by defining a plugin inside the module's src/Plugin/rest/resource directory.

The @RestResource annotation is used to define a custom REST resource plugin.

The class extends ResourceBase while implementing required methods.

HTTP methods are defined by implementing functions like get(), post(), patch(), and delete().

Validation is handled inside the method logic or using Drupal’s validation services before processing data.

Custom response codes are returned using ResourceResponse with the desired HTTP status code.

- **Security & Performance**

REST APIs are secured using proper authentication, permissions, HTTPS, and CSRF protection.

CSRF protection is implemented using Drupal's CSRF token service for unsafe HTTP methods.

Drupal REST responses are cacheable using cache metadata like tags, contexts, and max-age.

Cache contexts vary responses based on request conditions while cache tags invalidate related cached data.

Caching can be disabled by setting max-age to 0 in the response object.

- **Practical / Scenario-Based**

I would enable POST on the node resource, configure authentication, and allow required permissions for the mobile app role.

API versioning can be handled using custom routes or URL prefixes like /api/v1/.

File uploads are handled using the file entity REST endpoint with multipart/form-data requests.

REST issues are debugged using logs, watchdog, browser network tab, and tools like Postman.

Common REST errors include 403 permission issues and 415 unsupported media type, which are resolved by fixing permissions or headers.
