**1. How can mobile apps handle errors and timeouts when interacting with RESTful APIs?**

Mobile apps can handle errors and timeouts by:

* **Using Try-Catch Blocks**: Wrap API calls in try-catch blocks to handle exceptions gracefully.
* **Checking HTTP Status Codes**: After making a request, check the response status code (e.g., 404 for "Not Found", 500 for "Server Error") and handle each case accordingly.
* **Displaying Error Messages**: Show user-friendly error messages or retry options when something goes wrong.
* **Handling Timeouts**: Set a timeout duration for API calls and show a message or retry option if the call takes too long.

**2. What is API versioning, and why is it important in mobile app development?**

**API versioning** is the practice of managing changes to an API by creating different versions (e.g., v1, v2). It allows developers to introduce new features or improvements without breaking existing functionality for users on older versions.

* **Importance**: Ensures backward compatibility, providing a smooth user experience and preventing crashes in apps that rely on older versions of the API.

**3. What are the steps to integrate a RESTful API into a Flutter app?**

Steps to integrate a RESTful API into a Flutter app:

1. **Add the HTTP package**: Include http or another package in pubspec.yaml.
2. **Import the package**: Import the HTTP package into your Dart file.
3. **Make API Calls**: Use methods like http.get(), http.post() to make requests.
4. **Parse JSON Data**: Use dart:convert to parse the response data.
5. **Update UI**: Use Flutter widgets like ListView or Text to display the fetched data.
6. **Handle Errors**: Use try-catch blocks and check HTTP status codes to handle errors.

**4. Which packages in Flutter are commonly used to interact with RESTful APIs, and what are their differences?**

Common packages in Flutter to interact with RESTful APIs:

* **http**: A simple and widely used package for making HTTP requests. It provides basic functionalities like GET, POST, PUT, and DELETE requests.
* **Dio**: A more advanced and flexible HTTP client for Dart, providing features like interceptors, global configuration, request cancellation, and more. It is more suitable for complex use cases.
* **Retrofit**: A type-safe HTTP client built on top of Dio, allowing you to define APIs as interfaces and automatically generate HTTP requests. It's useful for more structured and maintainable codebases.

**Difference**: http is simpler and great for basic use cases, while Dio and Retrofit offer more advanced features for more complex applications.

**5. How do you handle authentication with RESTful APIs in Flutter apps (e.g., storing and using tokens)?**

To handle authentication with RESTful APIs in Flutter apps:

1. **Obtain a Token**: Make a login request to the API with user credentials and receive a token (e.g., JWT token).
2. **Store the Token**: Use packages like shared\_preferences to securely store the token on the device.
3. **Include Token in Requests**: Add the token to the headers of subsequent API requests using Authorization: Bearer <token>.
4. **Handle Token Expiry**: Check for token expiry and implement a refresh token mechanism to get a new token if needed.
5. **Logout**: Clear the stored token when the user logs out.