Remote Databases

1- RESTful APIs

- Representational State Transfer, is widely used architectural style for designing network applications
- Allows apps to communicate over HTTP
- Use GET, POST, PUT and DELETE

GET	Retrieves data from the server
POST	Sends new data to the server
PUT	Updates existing data
DELETE	Removes existing data

Example (How RESTful APIs works in a travel app)

Step 1: When user search for the flights the app sends the API request to the server using the GET command.

Step 2: Server Process the request

Step 3 : Server sends back the available results in JSON format

Step 4: The app then displays the flight options.

2- GraphQL

Offers more efficient way to communicate with the databases by allowing clients to

- Request only data they need
- Reduces bandwidth usage
- Improves app's performance

3- WebSockets

- Enables real-time communication
- Provide continuous connection

- Share data instantly with refreshing

Best practices for mobile developers

1- Optimize Network Usage

- Minimize data usage
- Caching frequently used data locally

2- Handle errors gracefully

- Inform users about issues
- Provide fallback options

3- Secure data transmission

- Use HTTPs
- Protecting sensitive data

4- Test connectivity

- Handle different network conditions (such as WiFi, mobile data and offline)
- Implement retry logic for failed requests

5- Monitor performance

- Use analytics tools to track API performance
- Optimize response times

Considerations for connecting mobile apps to remote databases

1- **Security**
- **Data protection:** Use SSL / TLS for secure communication.
- **User Authentication:** Implement Authentication methods like OAuth or JWT.

2- **Performance**
- **Speed:** Optimize API to reduce response times and minimize delays in server requests
- **Data Caching**: Stores frequently accessed data locally and Reduce repeated server requests.

3- **Scalability**
- **Growing with Demand:** Use Scalable Databases that can handle increased user demands

4- **Data Consistency**
- **Keep data aligned**: Ensure local data matches the remote database and sync offline data

5- **Network Reliability**
- **Handle Disconnections:** Implement retry logic
- **Offline functionality:** enables offline tasks and sync data when online

6- **Testing**
- **Full testing:** verifies that the entire system works together efficiently
- **Stress testing**: Test apps performance under heavy user load