Practical Task 2: Technical Research Report

1. The Python `requests` Module

The Python `requests` module is a popular third-party library used to send HTTP requests from Python code. It provides a simple and human-friendly way to interact with web servers, APIs, and web resources over the internet. With `requests`, developers can perform common HTTP operations such as GET (retrieving data), POST (submitting data), PUT (updating data), and DELETE (removing data) using straightforward function calls. For example, to fetch data from a web page, you can use `requests.get(url)`, and to submit data, you can use `requests.post(url, data=...)` or `requests.post(url, json=...)`[1] [2] [3].

The module returns a `Response` object that contains details about the server's reply, including the content, status code, headers, and more. This object allows easy access to the response data, whether it's plain text, JSON, or binary. The `requests` library also handles many complexities for you, such as encoding, sessions, cookies, and error handling, making it a go-to solution for working with HTTP in Python[1] [2] [3].

2. JSON and XML Data Formats

Both JSON (JavaScript Object Notation) and XML (eXtensible Markup Language) are widely used formats for storing and exchanging structured data between systems, especially in web applications and APIs. They each have unique characteristics, strengths, and weaknesses.

JSON

What it is used for:
JSON is a lightweight, text-based format designed for data interchange. It is commonly used for transmitting data between a server and a web application, especially in RESTful APIs and JavaScript-based environments[4] [5].

Advantages:

- Simpler and more concise syntax, making it easier to read and write[4] [5].
- Smaller file sizes, which leads to faster data transfer and lower bandwidth usage[4] [5].
- Native support in JavaScript and broad compatibility with many programming languages[4] [5].
- Faster parsing and serialization due to its straightforward structure[4] [5].

Disadvantages:

- Limited support for metadata and attributes compared to XML[5].
- No native support for comments, which can make documentation harder[5].

- Lacks built-in schema validation, though external tools like JSON Schema exist[4] [5].
- Can be less suitable for very complex or deeply nested data structures[4] [5].

XML

What it is used for:
XML is a markup language designed to encode documents and data in a format that is both human-readable and machine-readable. It is often used in enterprise applications, document storage, and scenarios requiring strict validation or metadata[4] [5].

Advantages:

- Highly extensible and supports custom tags for complex data representation[5].
- Strong support for metadata through attributes and namespaces[4] [5].
- Built-in validation capabilities using DTD and XSD schemas[4] [5].
- Widely accepted industry standard, ensuring broad compatibility[5].

Disadvantages:

- Verbose syntax leads to larger file sizes and slower transmission[4] [5].
- More complex to read, write, and parse compared to JSON[4] [5].
- Parsing XML requires more computational resources and specialized parsers[4] [5].
- Primarily text-based, with limited support for complex data types like arrays or binary data[5].

Feature	JSON (JavaScript Object Notation)	XML (eXtensible Markup Language)
Syntax	Key-value pairs, concise	Tag-based, verbose
Readability	Easy to read and write	More complex, harder to read
Metadata	Limited	Strong (attributes, namespaces)
Validation	Limited (via JSON Schema)	Strong (DTD, XSD)
File Size	Smaller	Larger
Parsing Speed	Fast	Slower
Use Cases	Web APIs, config files, data exchange	Document storage, config, complex data exchange

3. RESTful APIs

A RESTful API (Representational State Transfer Application Programming Interface) is a standardized way for different software systems to communicate over HTTP. RESTful APIs use standard HTTP methods—such as GET, POST, PUT, and DELETE—to perform operations on resources, which are typically represented as URLs. Each request is stateless, meaning all the information needed to process the request is included in that request, and the server does not retain client state between requests[6] [7].

RESTful APIs are widely used for web services and mobile apps, as they provide a simple and scalable way for clients and servers to interact. They are designed around resources, which are manipulated using standard HTTP verbs, and often exchange data in JSON or XML format[6] [7].

Advantages:

- Statelessness simplifies server design and improves scalability, as no client context is stored on the server[6] [7].
- Responses can be cached, which enhances performance and reduces server load[6] [7].
- Uniform interface using HTTP methods makes APIs easy to understand and use[6] [7].
- Client and server are separated, allowing for independent development and deployment[6] [7].
- Wide adoption and support across frameworks, tools, and programming languages[6].

Disadvantages:

- Clients may receive more data than needed (over-fetching) or may need to make multiple requests to get all required data (under-fetching)[6].
- Managing different API versions can become complex as applications evolve[6].
- Complex queries may require multiple endpoints or convoluted client logic[6].
- The fixed response structure may not always align with the needs of all clients, limiting flexibility[6].

References

[1] https://www.w3schools.com/python/module_requests.asp
[2] https://realpython.com/python-requests/
[3] https://mimo.org/glossary/python/requests-library
[4] https://apix-drive.com/en/blog/useful/xml-vs-json
[6] https://dev.to/vishalpawar1010/advantages-and-disadvantages-of-rest-api-365l
[5] https://leapcell.io/blog/xml-vs-json-a-comprehensive-comparison
[7] https://www.mailgun.com/blog/it-and-engineering/restful-api/