Cyber Security Part 2 of IA-1

1. **Explain the architecture of web services and the role of servers in hosting them.**

Answer:

Architecture of Web Services and the Role of Servers

Web services are applications that enable communication between different systems over the internet using standard protocols like HTTP, SOAP, or REST. Their architecture typically follows a **client-server model**, where:

1. **Client**: The requester that sends HTTP requests (e.g., a browser, mobile app, or another service).
2. **Web Server**: Hosts the web service, handles requests, and processes responses.
3. **Application Layer**: Business logic that processes the client's request.
4. **Database**: Stores and manages data that the service uses.
5. **Response**: The processed output sent back to the client.

Web servers play a crucial role by:

- Handling incoming requests and routing them.
- Serving static and dynamic content.
- Managing authentication, load balancing, and security.
- Enabling API communication between distributed systems.

2. Differentiate between RESTful and SOAP-based services.

Answer:

RESTful vs SOAP-Based Web Services

Feature	RESTful Services	SOAP-Based Services
Protocol	Uses HTTP methods (GET, POST, PUT, DELETE)	Uses XML-based messaging over HTTP, SMTP, or others
Message Format	JSON (mostly) or XML	XML (strict structure with WSDL)
Flexibility	Lightweight, scalable, and stateless	Heavy, strict contracts, and stateful support
Performance	Faster, less overhead	Slower due to XML processing
Security	Uses OAuth, JWT, HTTPS	WS-Security (built-in enterprise security)
Best Use Cases	Web, mobile apps, microservices	Banking, enterprise apps, legacy systems

3. Implement a simple HTTP-based web service using Flask or Node.js and deploy it on a server.

Answer:

app.py

```
from flask import Flask, jsonify, request

app = Flask(__name__)s

@app.route('/api/greet', methods=['GET'])

def greet():

    name = request.args.get('name', 'Guest')

    return jsonify({"message": f"Hello, {name}!"})

if __name__ == '__main__':

    app.run(host='0.0.0.0', port=5000, debug=True)
```

Command to run:

python app.py

How to test:

1. Using curl:

curl "http://localhost:5000/api/greet?name=John"

2. Go to browser and paste the link: "http://localhost:5000/api/greet?name=John"

Output:

```
totoro@ghibli:~/Roger/College/Sem_6/Cyber Security Lab/IA1-Part2$ python app.py
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5000
 * Running on http://192.168.1.6:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 104-436-044
127.0.0.1 - - [17/Mar/2025 22:52:45] "GET /api/greet?name=Jimit+Chavda HTTP/1.1" 200 -
```

```
totoro@ghibli:~/Roger/College/Sem_6/Cyber Security Lab/IA1-Part2$ curl "http://localhost:5000/api/greet?name=Jimit+Chavda"
{
  "message": "Hello, Jimit Chavda!"
}
```

Github Link:

Simple_Web_Service.git