

# **PRACTICAL LAB FILE**

**Subject: Cyber security Lab**

**Submitted By:**

**Name: Krish Patel**

**Roll No.: 22BCP322**

**Submitted To:**

**Utkarsh tiwari**

**Department of CSE**



**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

**SCHOOL OF TECHNOLOGY**

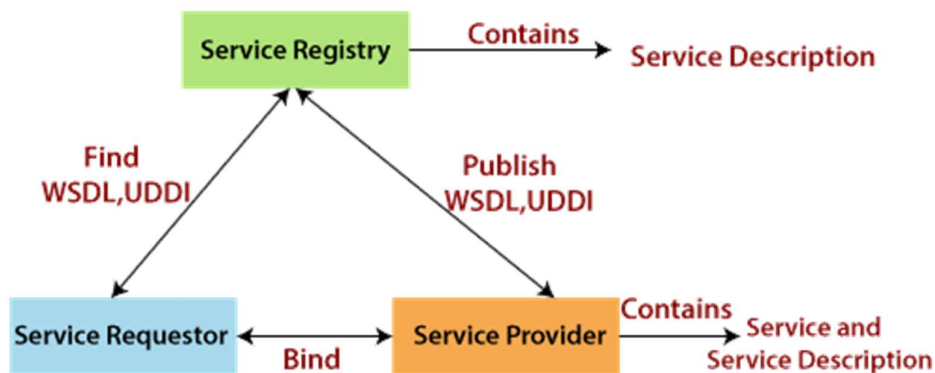
**PANDIT DEENDAYAL ENERGY UNIVERSITY, GANDHINAGAR**

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

## Architecture of Web Services

Web services are software systems designed to support interoperable machine-to-machine interaction over a network. They typically follow a client-server architecture, where the client sends a request to the server, and the server processes the request and sends back a response. The key components of web service architecture include:

1. Client: The application or system that initiates requests to the web service.
2. Server: The system that hosts the web service and processes client requests.
3. Protocols: Web services use standard protocols like HTTP/HTTPS, SOAP, REST, etc., for communication.
4. Data Formats: Data is exchanged in formats like XML, JSON, or plain text.
5. Service Description: Web services are often described using WSDL (Web Services Description Language) for SOAP-based services or OpenAPI/Swagger for RESTful services.



Web Service Roles, Operations and Artifacts

## Role of Servers in Hosting Web Services

Servers play a critical role in hosting web services. Their responsibilities include:

- Handling Requests: Servers listen for incoming client requests and process them.
- Business Logic Execution: They execute the logic required to generate responses.
- Data Storage and Retrieval: Servers interact with databases or other storage systems to fetch or store data.
- Security: Servers enforce authentication, authorization, and encryption to protect data.

- Scalability: Servers can be scaled horizontally (adding more servers) or vertically (increasing server resources) to handle increased traffic.
- 

## 2. RESTful vs. SOAP-based Web Services

Feature	RESTful Web Services	SOAP-based Web Services
Protocol	Uses HTTP methods (GET, POST, PUT, DELETE)	Uses XML-based SOAP protocol
Message Format	JSON, XML, YAML, or plain text	Strictly XML-based
Flexibility	Lightweight and flexible	Heavyweight with strict standards
Performance	Faster due to less overhead	Slower due to XML parsing
Security	Uses HTTPS, OAuth, JWT	Uses WS-Security (built-in security mechanisms)
State Management	Stateless (each request is independent)	Can be stateful (maintains session information)
Use Case	Web and mobile applications, APIs for microservices	Enterprise-level applications needing strict security and reliability

---

### 3. Implementing a Simple HTTP-Based Web Service Using Node.js

```
const express = require('express');
const app = express();
app.use(express.json());

let books = [
  { id: 1, title: '1984', author: 'George Orwell' },
  { id: 2, title: 'To Kill a Mockingbird', author: 'Harper Lee' },
];

app.get('/books', (req, res) => {
  res.json(books);
});

app.get('/books/:id', (req, res) => {
  const book = books.find(b => b.id === parseInt(req.params.id));
  if (!book) return res.status(404).json({ error: 'Book not found' });
  res.json(book);
});

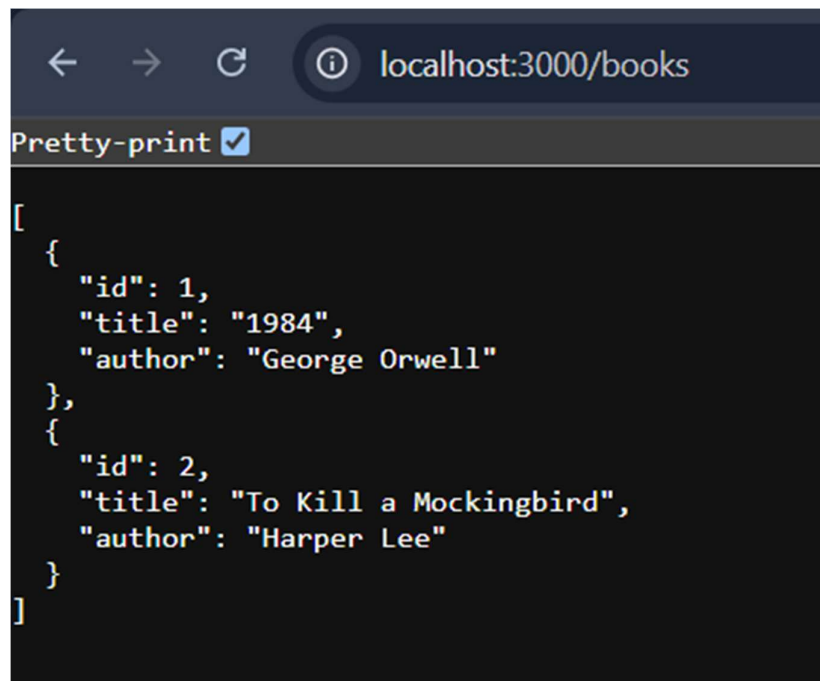
app.post('/books', (req, res) => {
  const newBook = req.body;
  books.push(newBook);
  res.status(201).json(newBook);
});

const PORT = 3000;
app.listen(PORT, () => {
  console.log(`Server is running on http://localhost:${PORT}`);
});
```

Output:



append /books after url to show list of books in json



A screenshot of a web browser window. The address bar shows 'localhost:3000/books'. Below the address bar is a toolbar with a 'Pretty-print' button and a checked checkbox. The main content area displays a JSON array of two book objects, formatted with syntax highlighting.

```
[
  {
    "id": 1,
    "title": "1984",
    "author": "George Orwell"
  },
  {
    "id": 2,
    "title": "To Kill a Mockingbird",
    "author": "Harper Lee"
  }
]
```