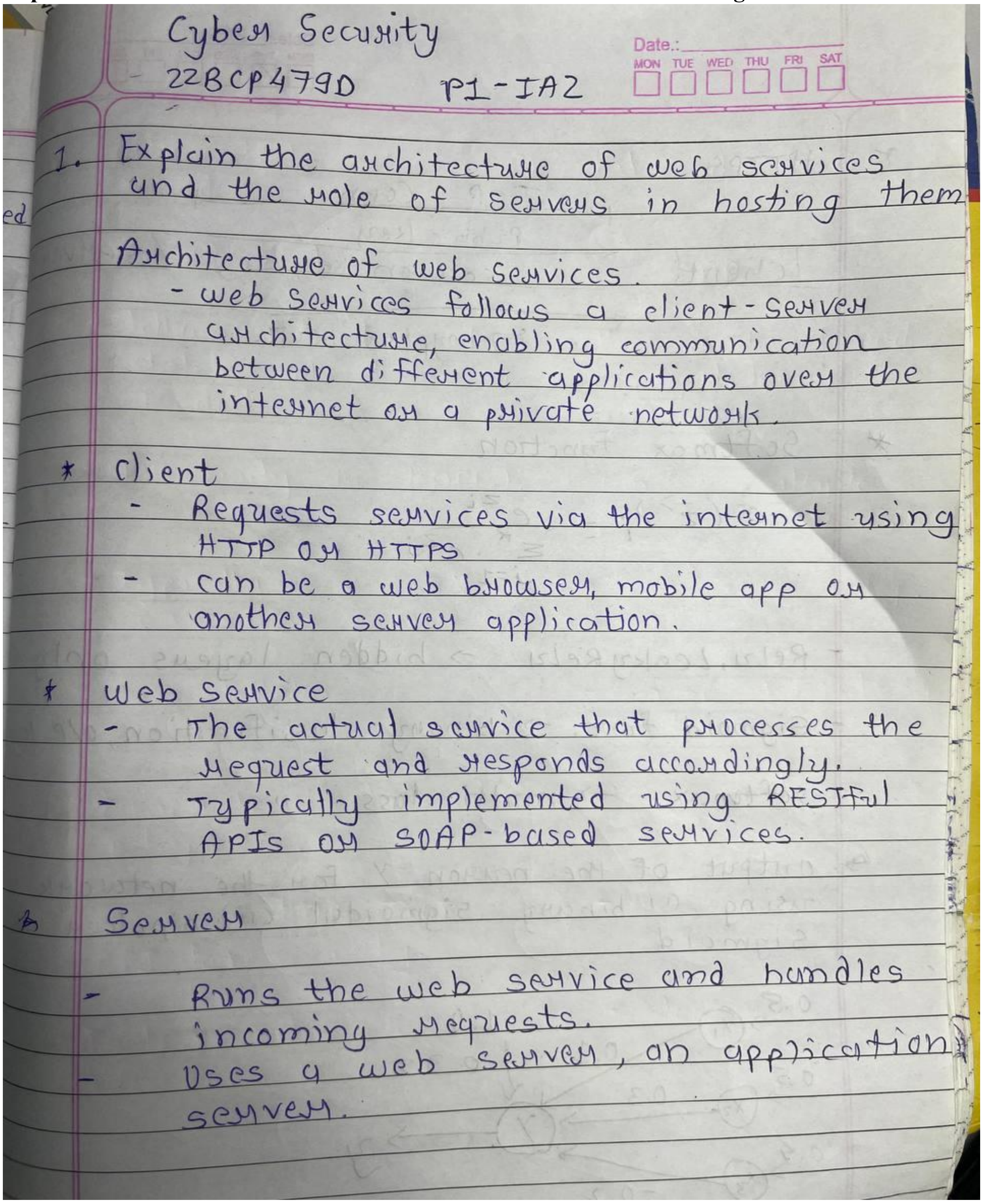
Unit 3 - Prerequisite Assignment (Part 1 of IA 2)

Krupal Lathiya
22BCP479D

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



Cyber Security
22BCP479D P1-IA2

Date.:
MON TUE WED THU FRI SAT

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

Architecture of web services.
- web services follows a client-server architecture, enabling communication between different applications over the internet or a private network.

* client
- Requests services via the internet using HTTP or HTTPS
- can be a web browser, mobile app or another server application.

* web service
- The actual service that processes the request and responds accordingly.
- Typically implemented using RESTful APIs or SOAP-based services.

* Server

- Runs the web service and handles incoming requests.
- Uses a web server, an application server.

* Database
  - stores and retrives data as required by the service.
  - Ex. MySQL, MongoDB.

* Network Infrastructure.
  - includes routers, firewalls, load balancers to manage traffic and security.

⇒ Roles of servers in hosting web services.

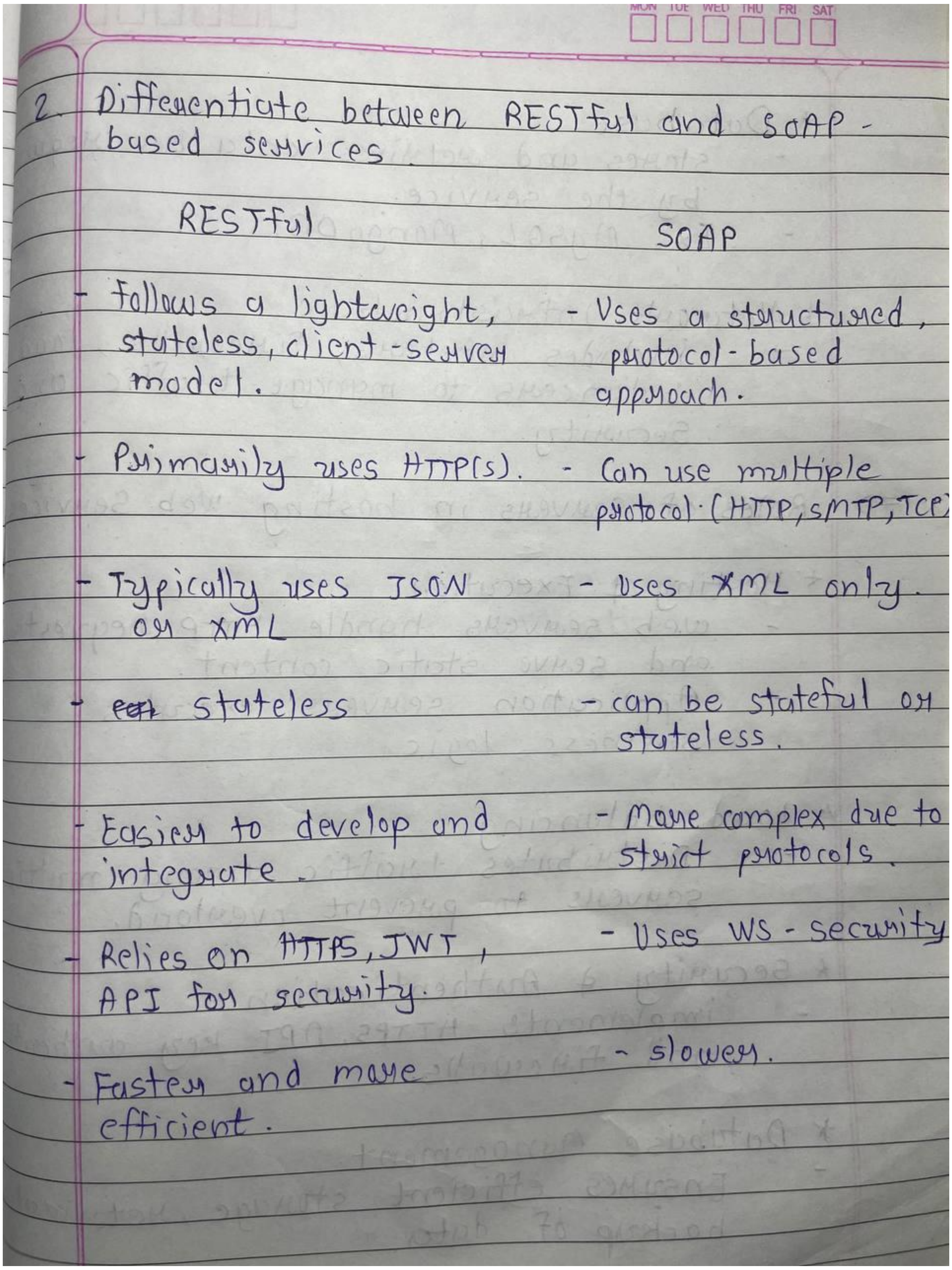
* Hosting & Execution.
  - web servers handle HTTP requests and serve static content.
  - Application servers executes business logic.

* load balancing
  - Distributes traffic among multiple servers to prevent overload.

* Security & Authentication.
  - implements HTTPS, API key authentication and firewalls.

* Database Management.
  - Ensures efficient storage, retrival, backup of data.

2) **Differentiate between RESTful and SOAP-based services.**



2. Differentiate between RESTful and SOAP-based services.

RESTful	SOAP
- Follows a lightweight, stateless, client-server model.	- Uses a structured, protocol-based approach.
- Primarily uses HTTP(S).	- Can use multiple protocol (HTTP, SMTP, TCP)
- Typically uses JSON or XML	- Uses XML only.
- ~~est~~ stateless	- can be stateful or stateless.
- Easier to develop and integrate.	- More complex due to strict protocols.
- Relies on HTTPS, JWT, API for security.	- Uses WS-security
- Faster and more efficient.	- slower.

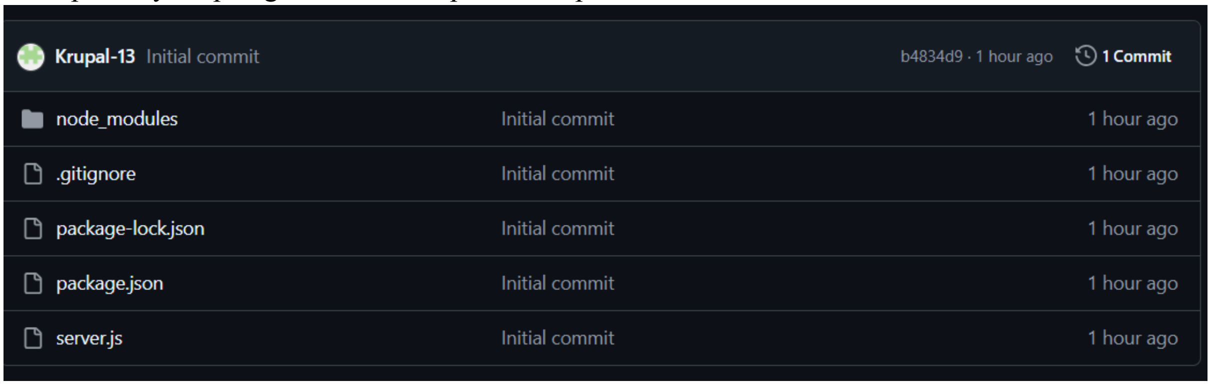
3) Implement a simple HTTP-based web service using Flask or Node.js and deploy it on a server.

Step 1) Create Simple node js app:

```
const express = require('express');
const app = express();
const port = process.env.PORT || 3000;


app.use(express.json());

app.get('/', (req, res) => {
   res.json({ message: "Welcome to the Node.js Web Service!" });
});

app.get('/users', (req, res) => {
   const users = [
      { id: 1, name: "Alice" },
      { id: 2, name: "Bob" }
   ];
   res.json(users);
});

app.listen(port, () => {
   console.log(`Server running at http://localhost:${port}`);
});
```

Step 2) Push it to GitHub:

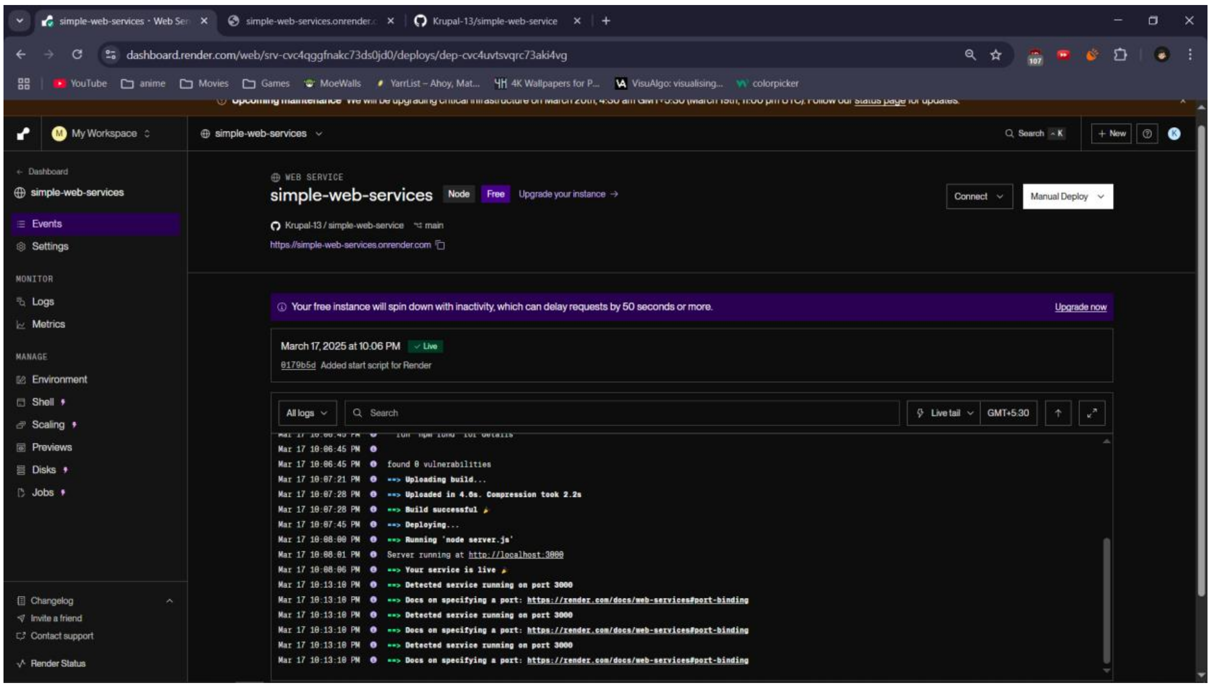
Git repository: https://github.com/Krupal-13/simple-web-service



Krupal-13 Initial commit   b4834d9 · 1 hour ago   1 Commit

node_modules	Initial commit	1 hour ago
.gitignore	Initial commit	1 hour ago
package-lock.json	Initial commit	1 hour ago
package.json	Initial commit	1 hour ago
server.js	Initial commit	1 hour ago

Step 3) Deploy on the Server: (Source: use Render)



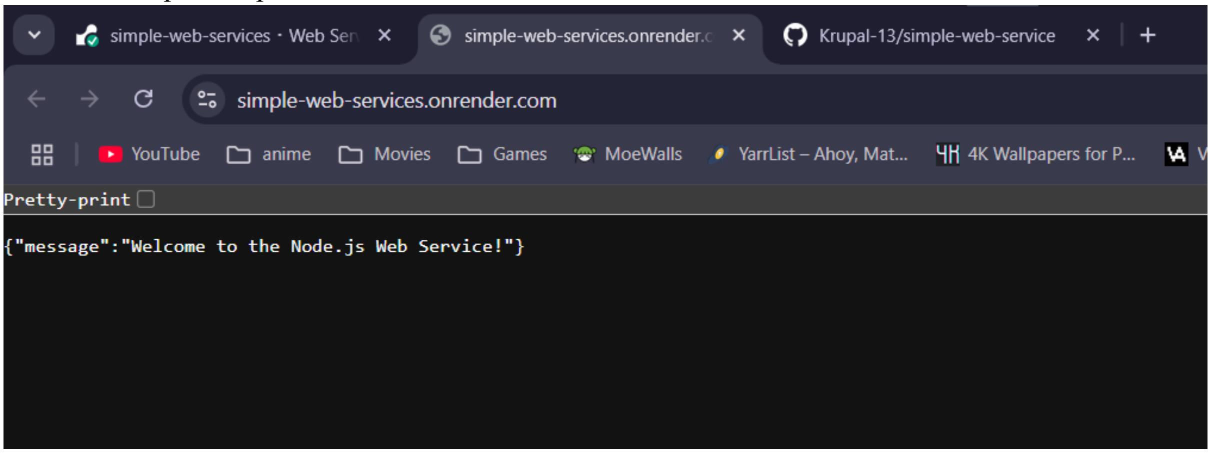
Your free instance will spin down with inactivity, which can delay requests by 50 seconds or more.

March 17, 2025 at 10:06 PM ✓ Live
0179b5d Added start script for Render

```
Mar 17 10:06:45 PM
Mar 17 10:06:45 PM   found 0 vulnerabilities
Mar 17 10:07:21 PM   ==> Uploading build...
Mar 17 10:07:28 PM   ==> Uploaded in 4.6s. Compression took 2.2s
Mar 17 10:07:28 PM   ==> Build successful 🎉
Mar 17 10:07:45 PM   ==> Deploying...
Mar 17 10:08:00 PM   ==> Running 'node server.js'
Mar 17 10:08:01 PM   Server running at http://localhost:3000
Mar 17 10:08:06 PM   ==> Your service is live 🎉
Mar 17 10:13:10 PM   ==> Detected service running on port 3000
Mar 17 10:13:10 PM   ==> Docs on specifying a port: https://render.com/docs/web-services#port-binding
Mar 17 10:13:10 PM   ==> Detected service running on port 3000
Mar 17 10:13:10 PM   ==> Docs on specifying a port: https://render.com/docs/web-services#port-binding
Mar 17 10:13:10 PM   ==> Detected service running on port 3000
Mar 17 10:13:10 PM   ==> Docs on specifying a port: https://render.com/docs/web-services#port-binding
```

Step 4) Run the Program:

OUTPUT: https://simple-web-services.onrender.com


simple-web-services · Web Ser
simple-web-services.onrender.c
Krupal-13/simple-web-service
simple-web-services.onrender.com
Pretty-print

```
{"message":"Welcome to the Node.js Web Service!"}
```