

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 51
Microservices Practical 9

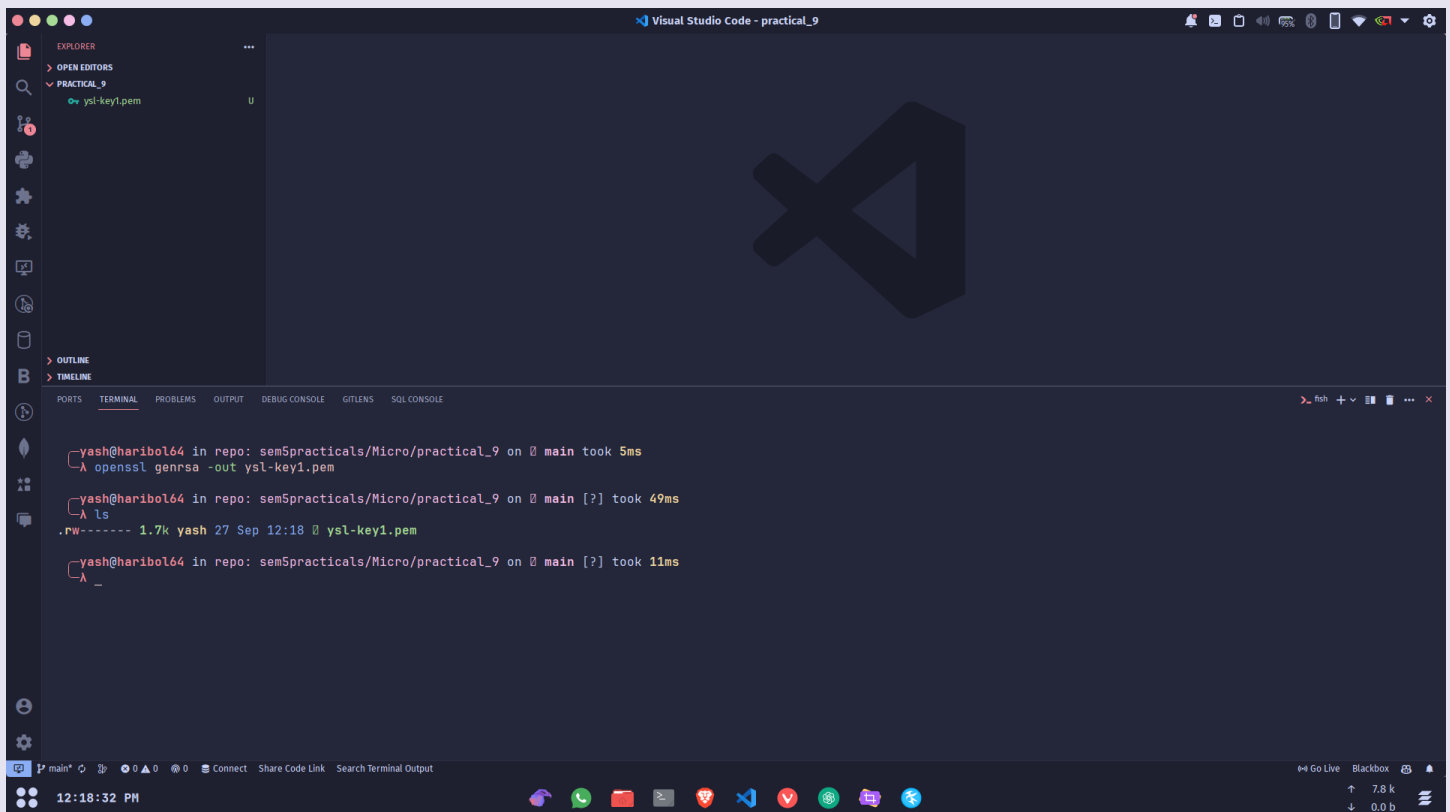
Scenario : Demonstrate the secured HTTP through SSL. Symbiosis Pvt Ltd makes NodeJS-based websites and deploys as well. At the time of deployment, websites require some authentication, and encryption policy for security purposes. Apply the following technique and secure said company's website while deploying :

Tasks, Code and Screenshots :

1. Generate a Public certificate with a public key and certificate

Command :

```
openssl genrsa -out ysl-key1.pem
```



The screenshot shows a Visual Studio Code window with a terminal open. The terminal output shows the command `openssl genrsa -out ysl-key1.pem` being executed successfully. The output indicates that the command took 5ms to execute. The terminal also shows the file `ysl-key1.pem` being created in the `practical_9` directory. The terminal output is as follows:

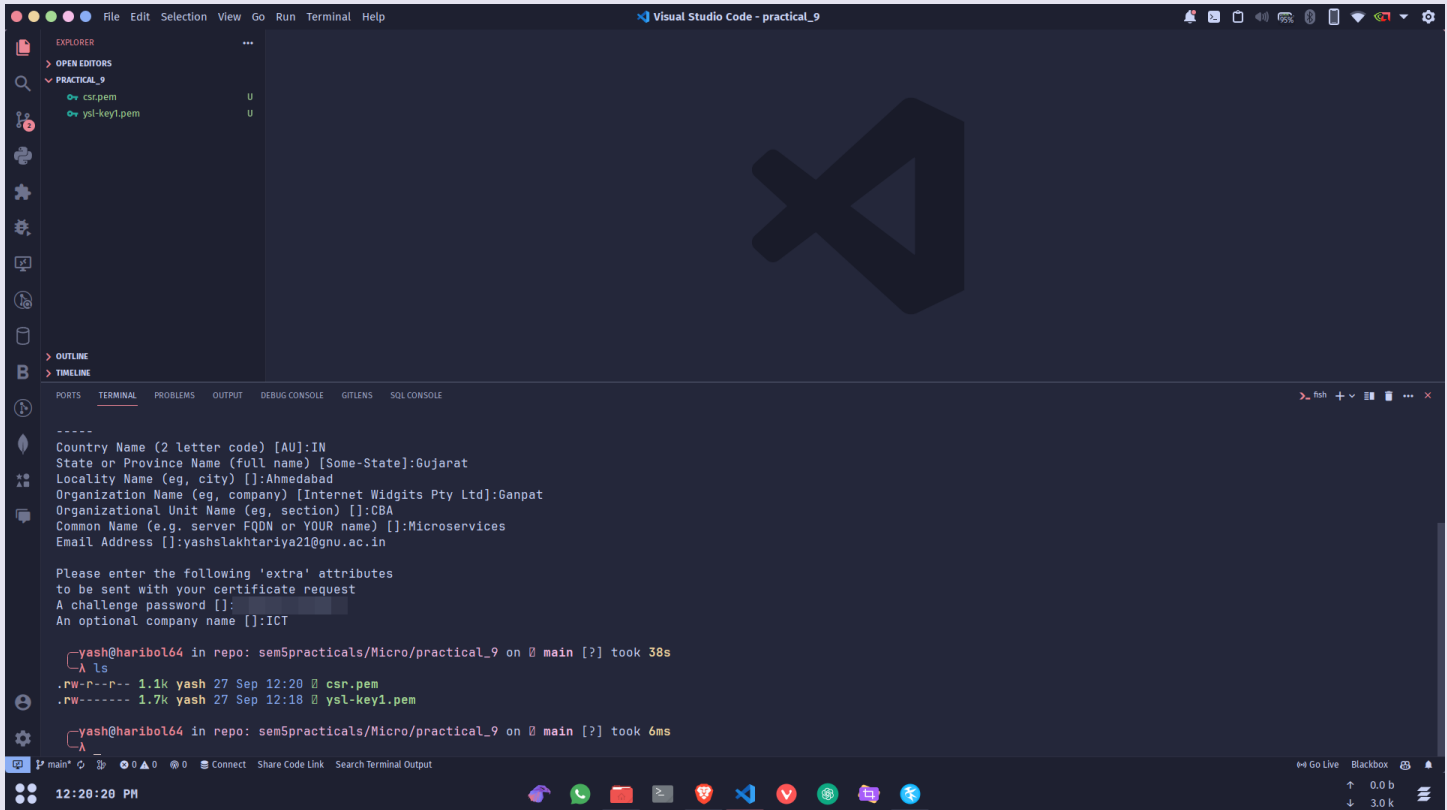
```
yash@haribol64 in repo: sem5practicals/Micro/practical_9 on 0 main took 5ms
λ openssl genrsa -out ysl-key1.pem

yash@haribol64 in repo: sem5practicals/Micro/practical_9 on 0 main [?] took 49ms
λ ls
.PW----- 1.7k yash 27 Sep 12:18 0 ysl-key1.pem

yash@haribol64 in repo: sem5practicals/Micro/practical_9 on 0 main [?] took 11ms
λ _
```

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 51
Microservices Practical 9

```
openssl req -new -key ysl-key1.pem -out csr.pem
```



```
Visual Studio Code - practical_9

EXPLORER
  OPEN EDITORS
    PRACTICAL_9
      csr.pem
      ysl-key1.pem

TERMINAL
  -----
  Country Name (2 letter code) [AU]:IN
  State or Province Name (full name) [Some-State]:Gujarat
  Locality Name (eg, city) []:Ahmedabad
  Organization Name (eg, company) [Internet Widgits Pty Ltd]:Ganpat
  Organizational Unit Name (eg, section) []:CBA
  Common Name (e.g. server FQDN or YOUR name) []:Microservices
  Email Address []:yashslakhtariya21@gnu.ac.in

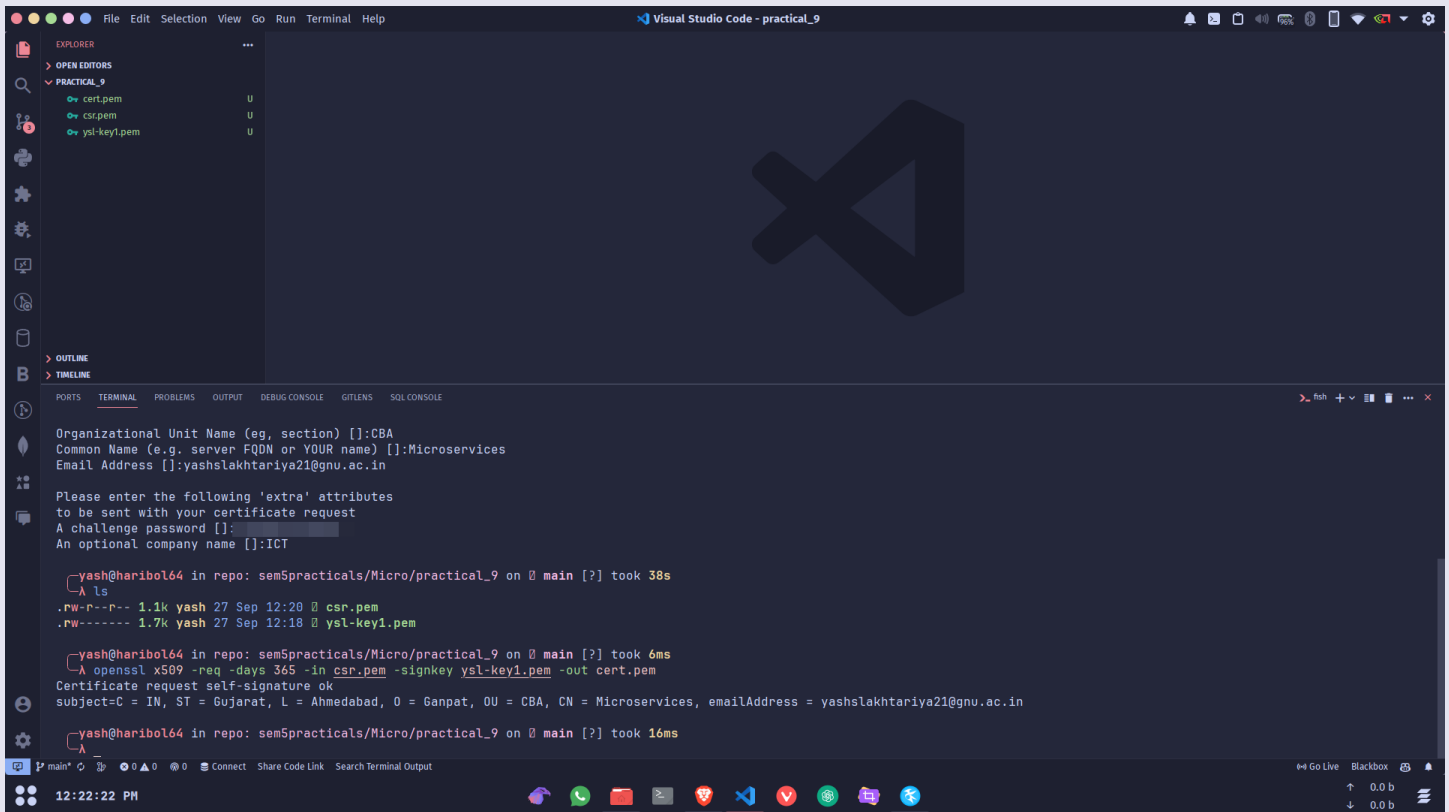
  Please enter the following 'extra' attributes
  to be sent with your certificate request
  A challenge password []:
  An optional company name []:ICT

  yash@haribol64 in repo: sem5practicals/Micro/practical_9 on  main (?) took 38s
  ^C
  .rw-r--r-- 1.1k yash 27 Sep 12:20  csr.pem
  .rw----- 1.7k yash 27 Sep 12:18  ysl-key1.pem

  yash@haribol64 in repo: sem5practicals/Micro/practical_9 on  main (?) took 6ms
```

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 51
Microservices Practical 9

```
openssl x509 -req -days 365 -in csr.pem -signkey ysl-key1.pem -out cert.pem
```



The screenshot shows the Visual Studio Code interface with a terminal window open. The terminal displays the following output:

```
Organizational Unit Name (eg, section) []:CBA
Common Name (e.g. server FQDN or YOUR name) []:Microservices
Email Address []:yashslakhtariya21@gnu.ac.in

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:ICT

yash@haribol64 in repo: sem5practicals/Micro/practical_9 on 0 main (?) took 38s
ls
-rw-r--r-- 1.1k yash 27 Sep 12:20 0 csr.pem
-rw----- 1.7k yash 27 Sep 12:18 0 ysl-key1.pem

yash@haribol64 in repo: sem5practicals/Micro/practical_9 on 0 main (?) took 6ms
openssl x509 -req -days 365 -in csr.pem -signkey ysl-key1.pem -out cert.pem
Certificate request self-signature ok
subject=C = IN, ST = Gujarat, L = Ahmedabad, O = Ganpat, OU = CBA, CN = Microservices, emailAddress = yashslakhtariya21@gnu.ac.in

yash@haribol64 in repo: sem5practicals/Micro/practical_9 on 0 main (?) took 16ms
```

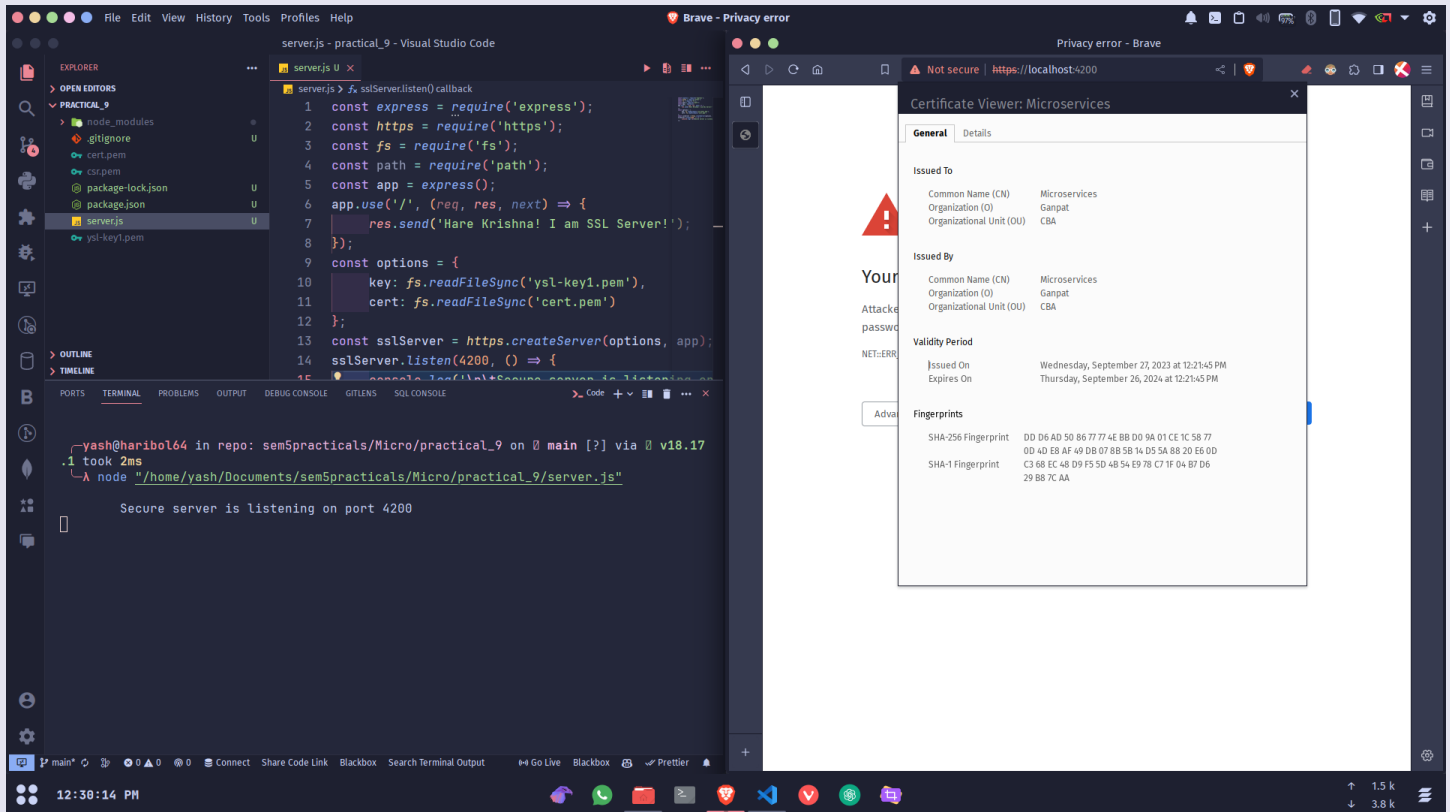
The terminal output shows the successful execution of the openssl command to generate a certificate. The subject information is: C = IN, ST = Gujarat, L = Ahmedabad, O = Ganpat, OU = CBA, CN = Microservices, emailAddress = yashslakhtariya21@gnu.ac.in.

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 51
Microservices Practical 9

2. Connect it with the website using the HTTPS library

```
const express = require('express');
const https = require('https');
const fs = require('fs');
const path = require('path');
const app = express();
app.use('/', (req, res, next) => {
res.send('Hare Krishna! I am SSL Server!');
});
const options = {
key: fs.readFileSync('ssl-key1.pem'),
cert: fs.readFileSync('cert.pem')
};
const sslServer = https.createServer(options, app);
sslServer.listen(4200, () => {
console.log('\n\tSecure server is listening on port 4200');
});
```

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 51
Microservices Practical 9



3. Secure application using Username and password.

```
// Requiring module
const express = require("express");

const app = express();

function authentication(req, res, next) {
  var authheader = req.headers.authorization;
  console.log(req.headers);

  if (!authheader) {
```

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 51
Microservices Practical 9

```
var err = new Error('You are not authenticated!');
res.setHeader('WWW-Authenticate', 'Basic');
err.status = 401;
return next(err);
}

var auth = new Buffer.from(authheader.split(' ')[1],
'base64').toString().split(':');
var user = auth[0];
var pass = auth[1];

if (user === 'admin' && pass === 'password') {
  // If Authorized user
  next();
} else {
  var err = new Error('You are not authenticated!');
  res.setHeader('WWW-Authenticate', 'Basic');
  err.status = 401;
  return next(err);
}
}

// First step is the authentication of the client
app.use(authentication);
//app.use(express.static(path.join(__dirname, 'public')));
```

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 51
Microservices Practical 9

```
app.get('/protected', (req, res) => {  
  res.send('I can be reached only using an authorised api key.');
```



```
});  
  
// Server setup  
app.listen(4200, () => {  
  console.log("Server is Running");  
});
```

The screenshot displays a development environment with two main windows. On the left, Visual Studio Code is open with a file named `secureapi.js` in the `practical_9` directory. The code defines an Express.js application with a `/protected` endpoint that requires an 'authorization' header. The server is configured to listen on port 4200. The terminal at the bottom shows the server's output, including the request headers from Postman: `authorization: 'Basic YWRtaW46cGZlc3dvcmQ='`, `user-agent: 'PostmanRuntime/7.32.3'`, and `accept-encoding: 'gzip, deflate, br'`. On the right, the Postman application is open, showing a workspace for `http://localhost:4200/protected`. A GET request is configured with a Basic Auth header, using the username `admin` and password `password`. The response body shows the text `I can be reached only using an authorised api key.` with a status of 200 OK.

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 51
Microservices Practical 9

4. Secure your REST APIs using the Bearer token technique.

```
const express = require("express");
const jwt = require("jsonwebtoken");

const app = express();
app.use(express.json());

app.post("/api/login", (req, res) => {
  //you can do this either synchronously or asynchronously
  //if synhronously, you can set a variable to jwt sign and pass it into the
  payload with secret key
  //if async => call back

  //Mock user
  const user = {
    id: Date.now(),
    userEmail: "yashlakhtariya1010@gmail.com",
    password: "harekrishna",
  };

  //send abpve as payload
  jwt.sign(
    { user },
    "mysecret",
```


Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 51
Microservices Practical 9

```
{
  // expiresIn: "10h" // it will be expired after 10 hours
  //expiresIn: "20d" // it will be expired after 20 days
  //expiresIn: 120 // it will be expired after 120ms
  expiresIn: "3600s", // it will be expired after 120s
},
(err, token) => {
  res.json({
    token,
  });
}
);
});

app.get("/api/profile", verifyToken, (req, res) => {
  jwt.verify(req.token, "mysecret", (err, authData) => {
    if (err) res.sendStatus(403);
    else {
      res.json({
        message: "Welcome to Profile",
        userData: authData,
      });
    }
  });
});
```

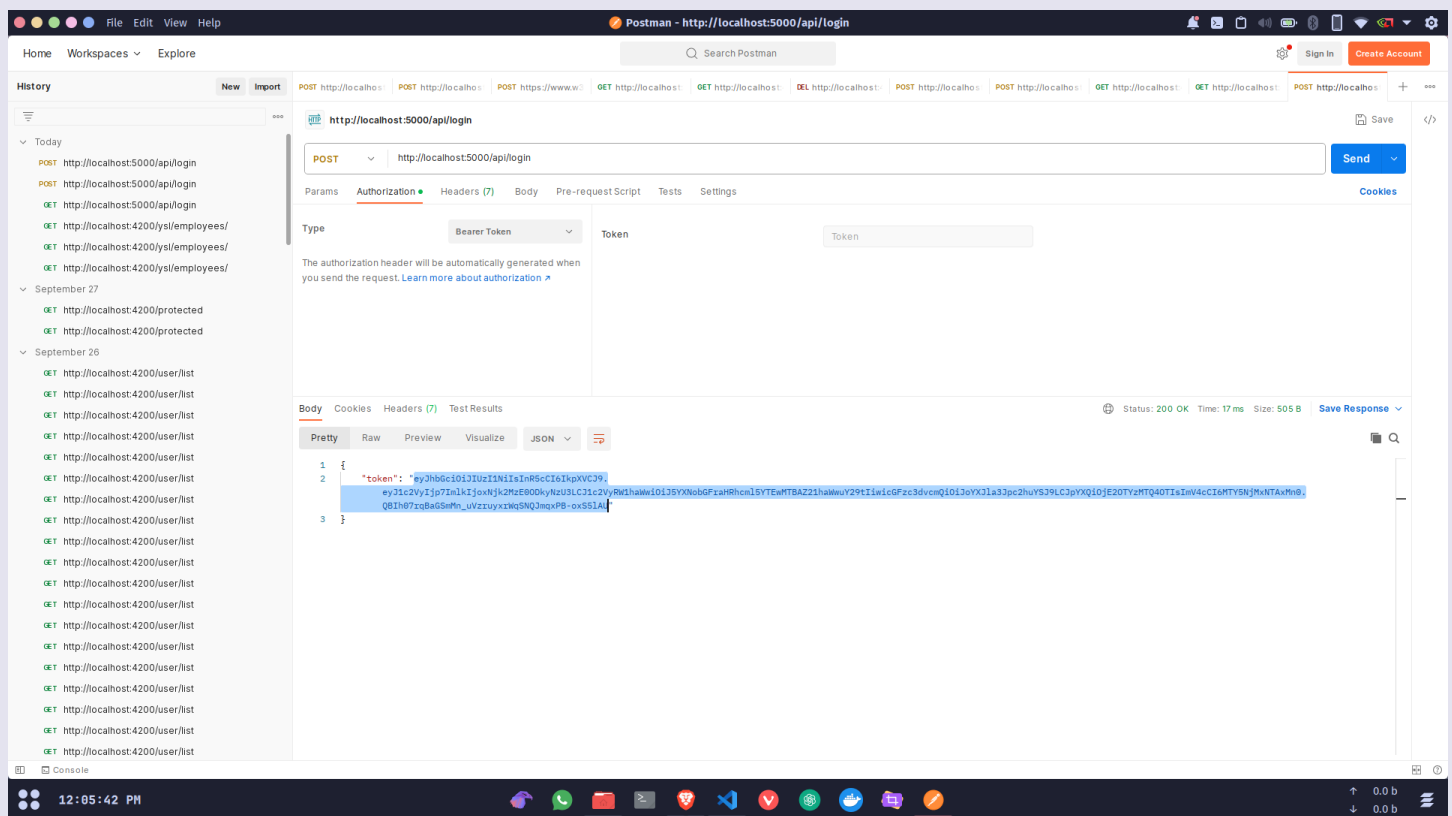
Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 51
Microservices Practical 9

```
});  
  
//Verify Token  
function verifyToken(req, res, next) {  
  //Auth header value = > send token into header  
  const bearerHeader = req.headers["authorization"];  
  //check if bearer is undefined  
  if (typeof bearerHeader !== "undefined") {  
    //split the space at the bearer  
    const bearer = bearerHeader.split(" ");  
    //Get token from string  
    const bearerToken = bearer[1];  
    //set the token  
    req.token = bearerToken;  
    //next middleware  
    next();  
  } else {  
    //Forbidden  
    res.sendStatus(403);  
  }  
}  
  
app.listen(5000, (err) => {  
  if (err) {
```

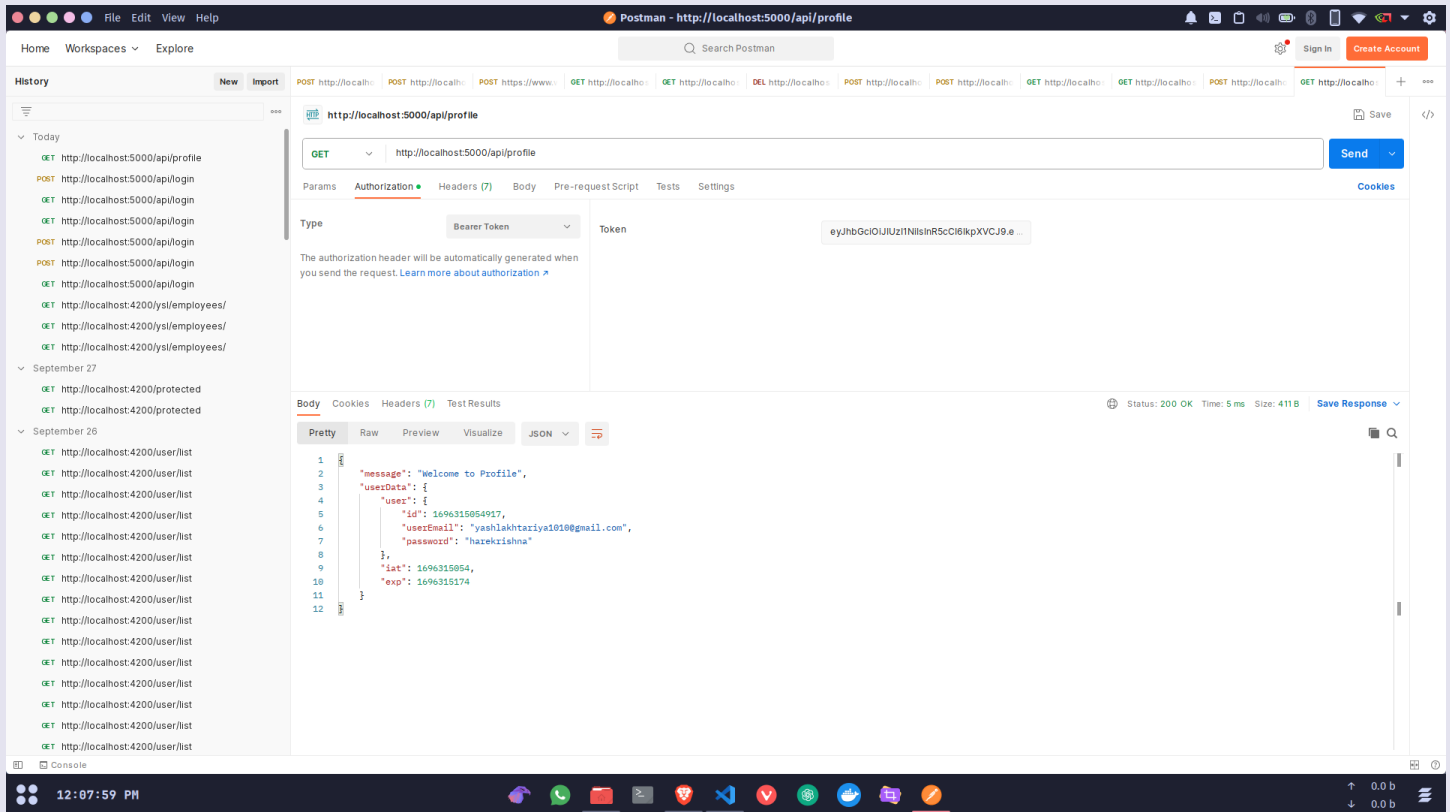
Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 51
Microservices Practical 9

```
console.log(err);
}

console.log("\n\tServer Started on PORT 5000");
});
```



Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 51
Microservices Practical 9



5. Secure your REST APIs created in previous practicals using username and password

```
const express = require("express");
const bodyParser = require("body-parser");
const cors = require("cors");
const app = express();
app.use(cors());

app.use(bodyParser.urlencoded({ extended: false }));
app.use(bodyParser.json());
let port = 4200;
```

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 51
Microservices Practical 9

```
function authentication(req, res, next) {  
  var authheader = req.headers.authorization;  
  console.log(req.headers);  
  
  if (!authheader) {  
    var err = new Error('You are not authenticated!');  
    res.setHeader('WWW-Authenticate', 'Basic');  
    err.status = 401;  
    return next(err);  
  }  
  
  var auth = new Buffer.from(authheader.split(' ')[1],  
    'base64').toString().split(':');  
  var user = auth[0];  
  var pass = auth[1];  
  
  if (user === 'yash' && pass === 'harekrishna') {  
    // If Authorized user  
    next();  
  } else {  
    var err = new Error('You are not authenticated!');  
    res.setHeader('WWW-Authenticate', 'Basic');  
    err.status = 401;  
    return next(err);  
  }  
}
```

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 51
Microservices Practical 9

```
}  
  
}  
  
let emps = [  
  {  
    id: "1",  
    name: "Madhav",  
    email: "madhav@haribol.com",  
    phone: "1234567890",  
    designation: "CEO",  
  },  
  {  
    id: "2",  
    name: "Gopal",  
    email: "gopal@haribol.com",  
    phone: "0123456789",  
    designation: "HR",  
  },  
  {  
    id: "3",  
    name: "Damodar",  
    email: "damodar@haribol.com",  
    phone: "9876543210",  
    designation: "Finance",  
  },  
]
```

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 51
Microservices Practical 9

```
},  
  
{  
  id: "4",  
  name: "Keshav",  
  email: "keshav@haribol.com",  
  phone: "9630147852",  
  designation: "HR",  
},  
];  
  
app.use(authentication);  
  
app.get("/", (req, res) => {  
  res.send("Welcome to YSL Company REST API!");  
});  
  
app.get("/ysl/employees", (req, res) => {  
  res.send(emps);  
});  
  
app.get("/ysl/employees/:id", (req, res) => {  
  const emp = emps.find(({ id }) => id === req.params.id);  
  
  if (!emp) res.status(404).send("Not found!");  
});
```

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 51
Microservices Practical 9

```
res.send(emp);  
});  
  
app.post("/ysl/employees", (req, res) => {  
  let emp = {  
    id: req.body.id,  
    name: req.body.name,  
    email: req.body.email,  
    phone: req.body.phone,  
    dept: req.body.dept,  
  };  
  emps.push(emp);  
  res.send(emp);  
});  
  
app.put("/ysl/employees/:id", (req, res) => {  
  let emp = emps.find(({ id }) => id === req.params.id);  
  if (!emp) res.status(404).send("Not found!");  
  emp.id = req.body.id;  
  emp.name = req.body.name;  
  emp.email = req.body.email;  
  emp.ln = req.body.ln;  
  emp.phone = req.body.phone;  
  emp.designation = req.body.designation;
```


Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 51
Microservices Practical 9

```
res.send(emp);  
});  
  
app.delete("/ysl/employees/:id", (req, res) => {  
  const emp = emps.find(({ id }) => id === req.params.id);  
  if (!emp) res.status(404).send("Not found!");  
  
  const index = emps.indexOf(emp);  
  emps.splice(index, 1);  
  
  res.send(emp);  
});  
  
app.listen(port, () => {  
  console.log(`\nThe server is running on port ${port}\n`);  
});
```

Name - Yash Lakhtariya

Enrollment number - 21162101012

Branch - CBA Batch - 51

Microservices Practical 9

The screenshot shows a development environment with Visual Studio Code on the left and Postman on the right. In VS Code, the file `securemyapi.js` is open, showing an Express.js application with a basic authentication middleware. The code includes `require('express')`, `require('body-parser')`, `require('cors')`, and `express()`. The `authentication` function checks for an `authorization` header in the request. The server is running on port 4200. The terminal output shows the command `node "/home/yash/Documents/sem5practicals/Micro/practical_9/securemyapi.js"` and the message "The server is running on port 4200".

Postman is configured with a GET request to `http://localhost:4200/ysl/employees/`. The response status is `401 Unauthorized` with a message: "Error: You are not authenticated!". The response body is an HTML error page.

The screenshot shows the same development environment as the previous one. In VS Code, the code is the same. The terminal output shows the command `node "/home/yash/Documents/sem5practicals/Micro/practical_9/securemyapi.js"` and the message "The server is running on port 4200".

Postman is configured with a GET request to `http://localhost:4200/ysl/employees/`. The response status is `200 OK` with a message: "The authorization header will be automatically generated when you send the request". The response body is a JSON array of employee data:

```
[{"id": "1", "name": "Madhav", "email": "madhav@haribol.com", "phone": "1234567890", "designation": "CEO"}, {"id": "2", "name": "Gopal", "email": "gopal@haribol.com", "phone": "8123456789", "designation": "HR"}, {"id": "3", "name": "Damodar", "email": "damodar@haribol.com", "phone": "9876543210", "designation": "Finance"}]
```