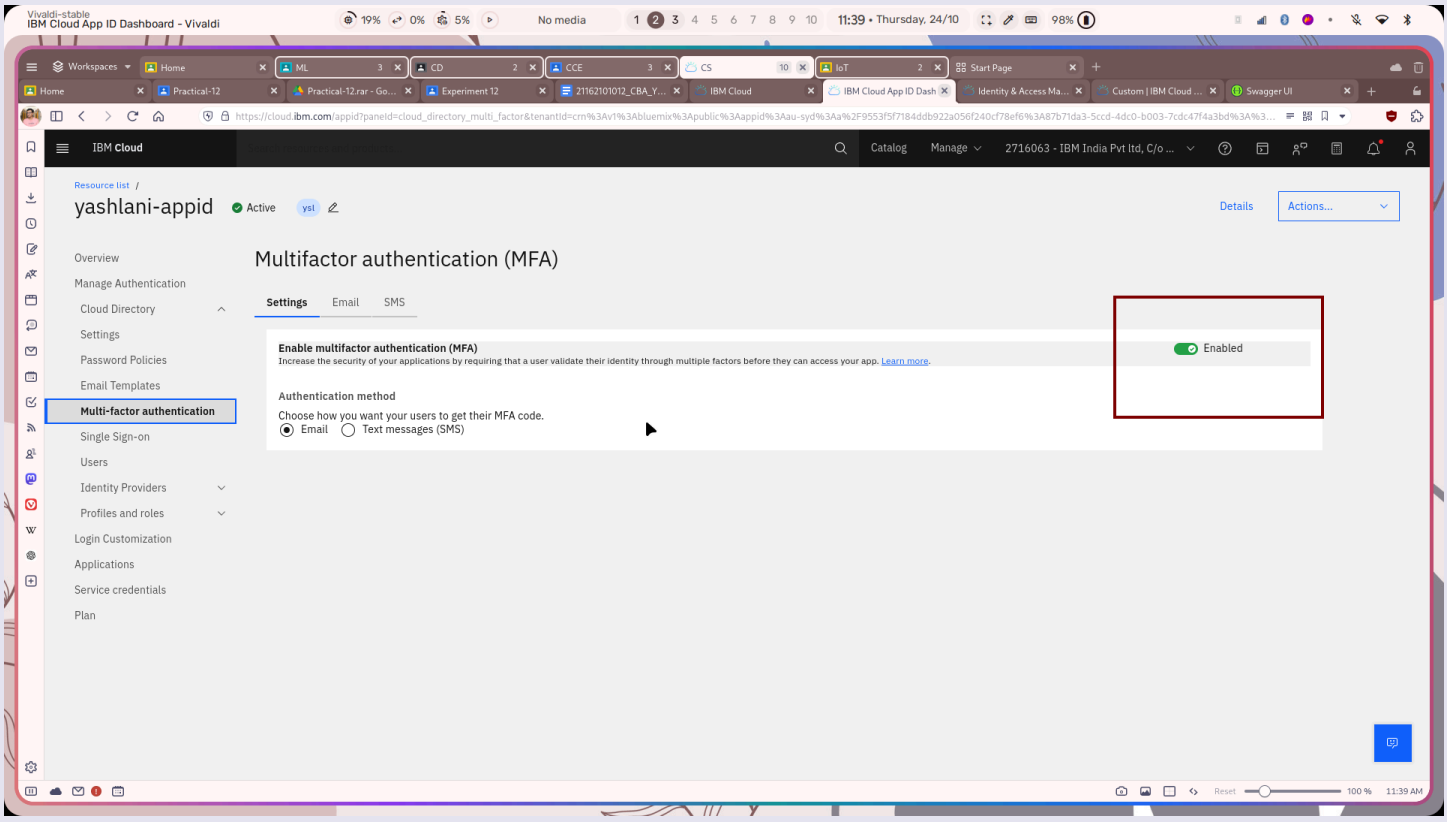
Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 71
CS Practical 12

Aim : You are tasked with setting up a secure authentication system for an enterprise application hosted on IBM Cloud. The application should enforce MFA for added security and integrate with a custom identity provider to accommodate users from a legacy system. Also work on the process of regenerating access tokens using refresh tokens.

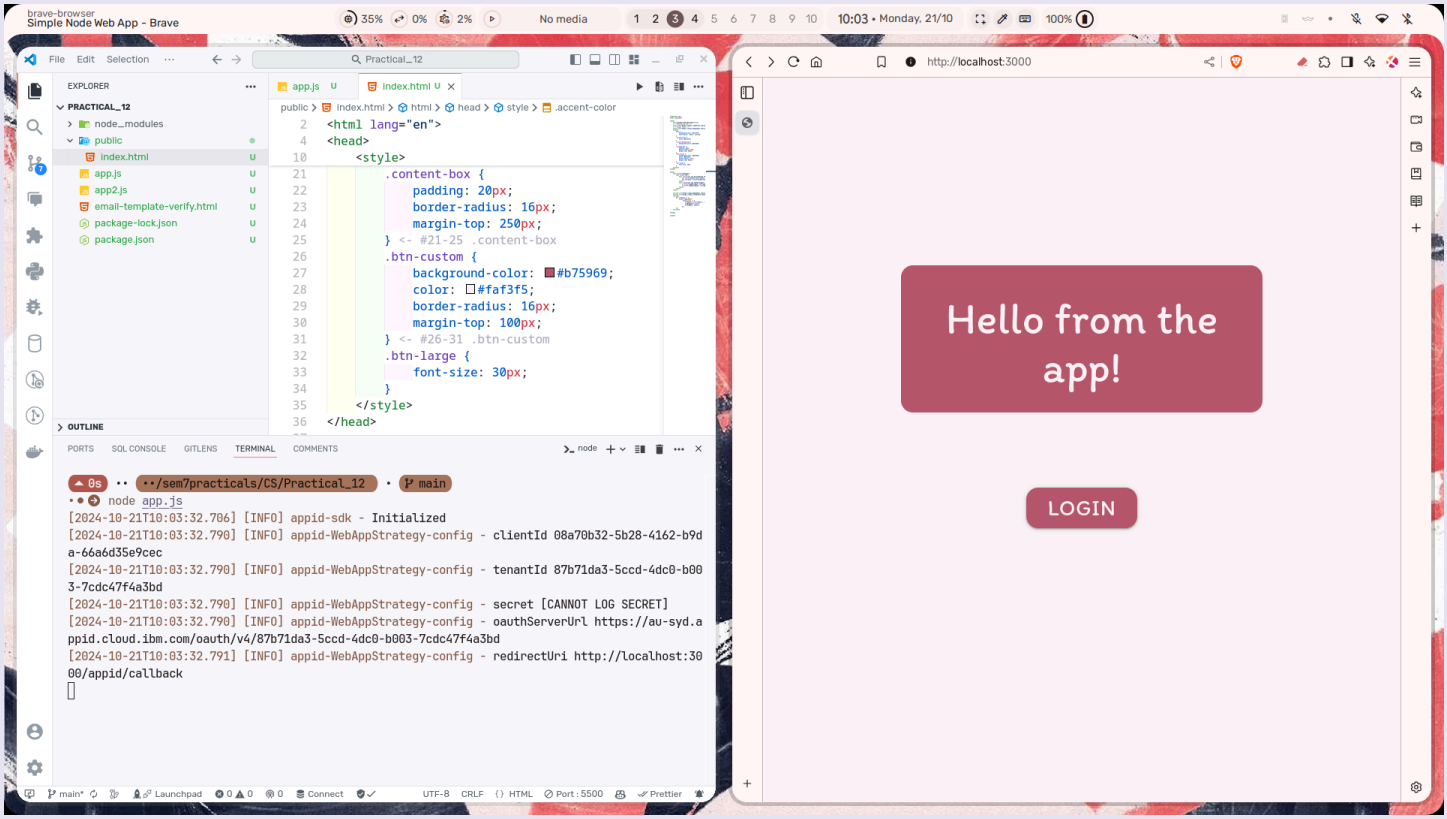
Steps and Screenshots :

1. Enable MFA from Cloud directory settings of App ID instance

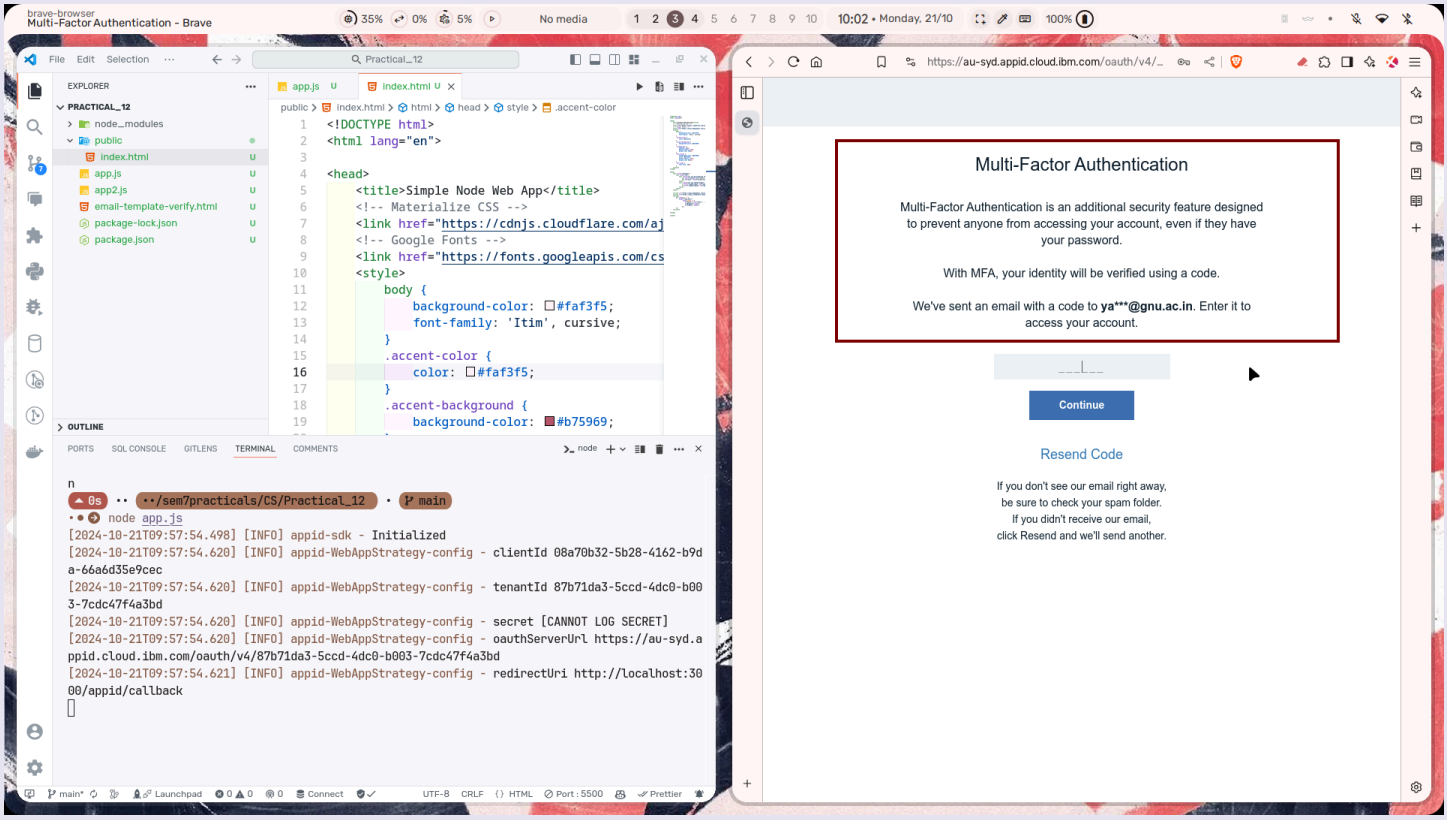


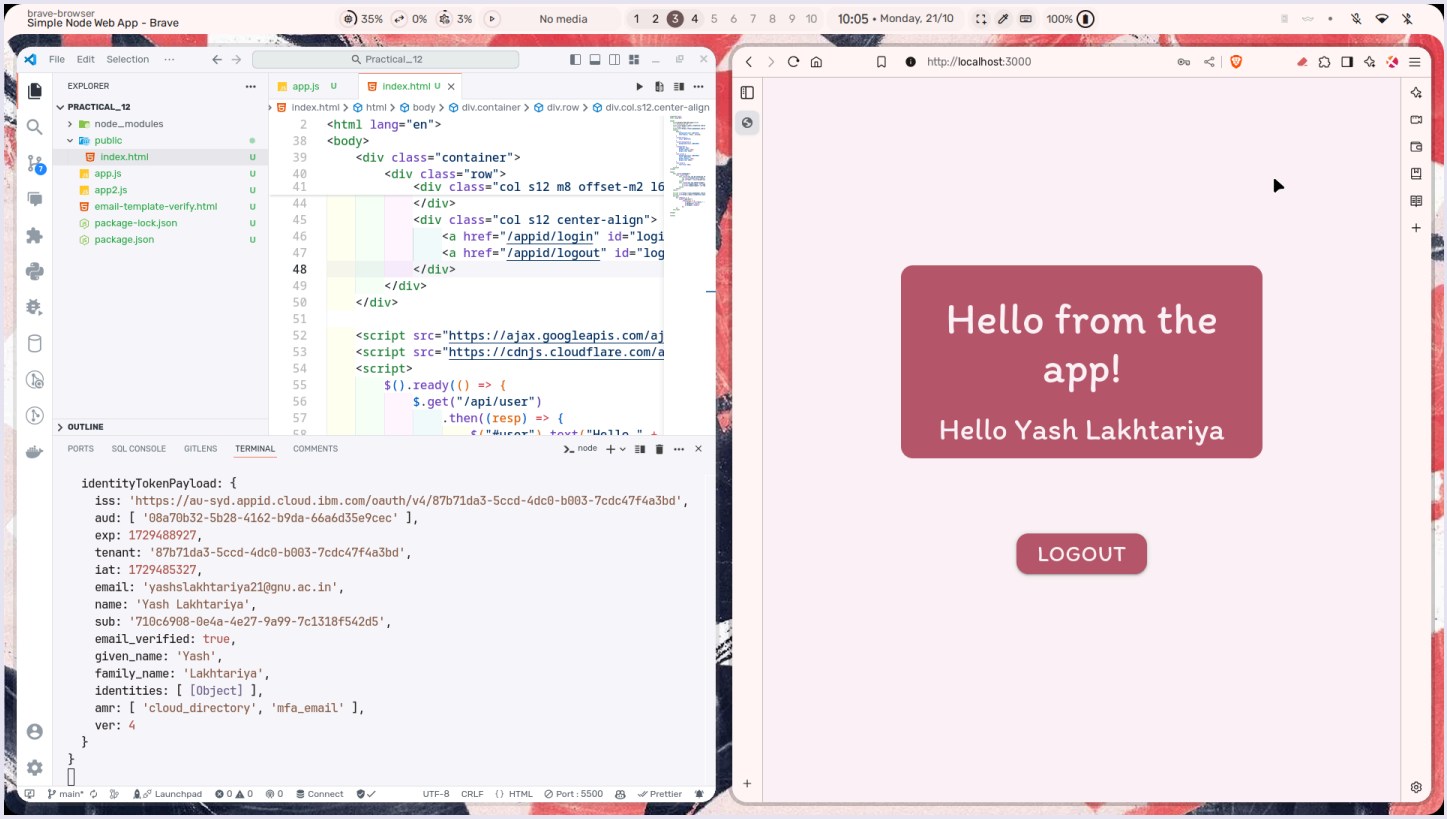
Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 71
CS Practical 12

2. Check if MFA works



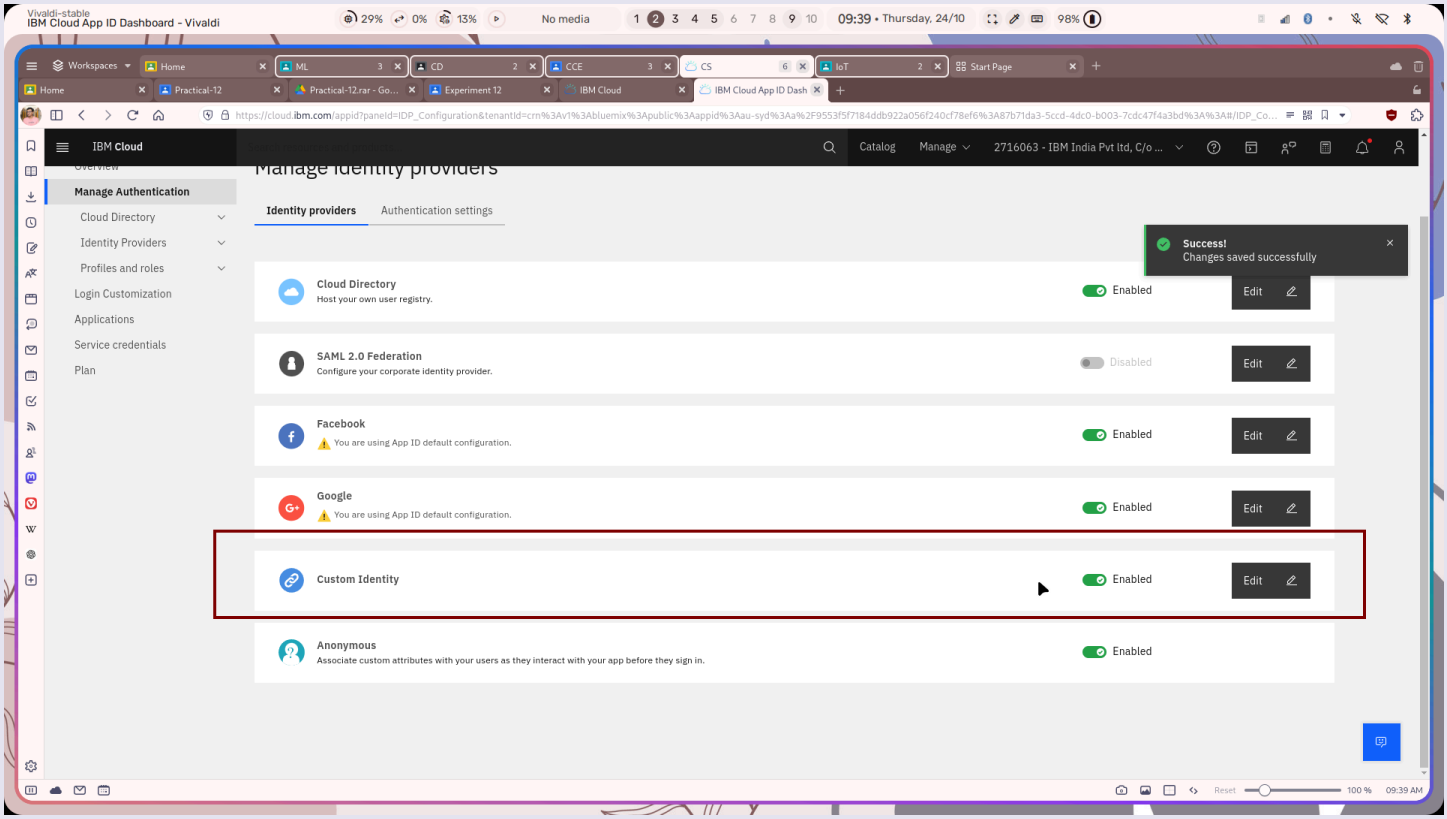
Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 71
CS Practical 12



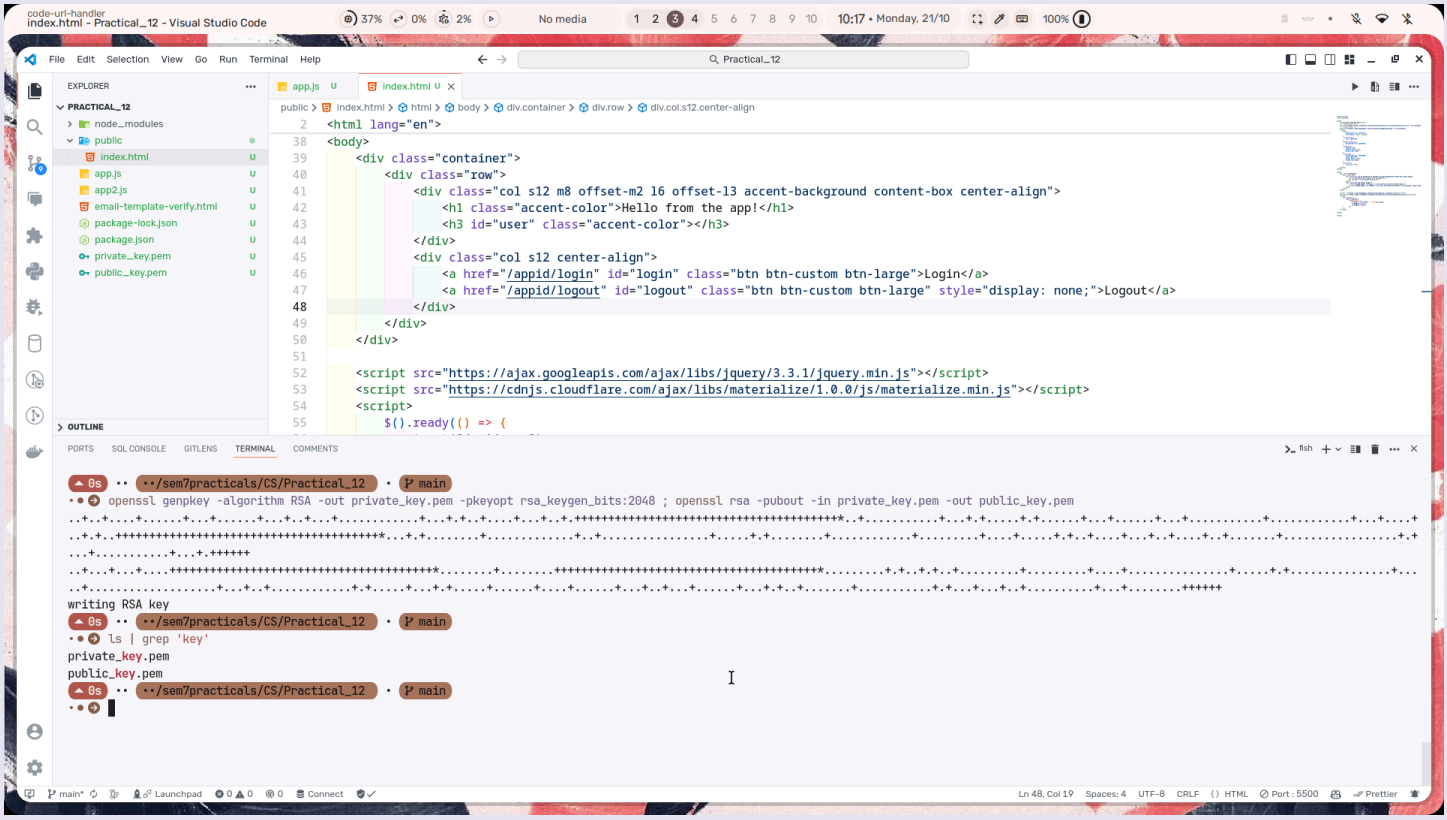


Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 71
CS Practical 12

3. Generate ssl keys to use with custom identity provider

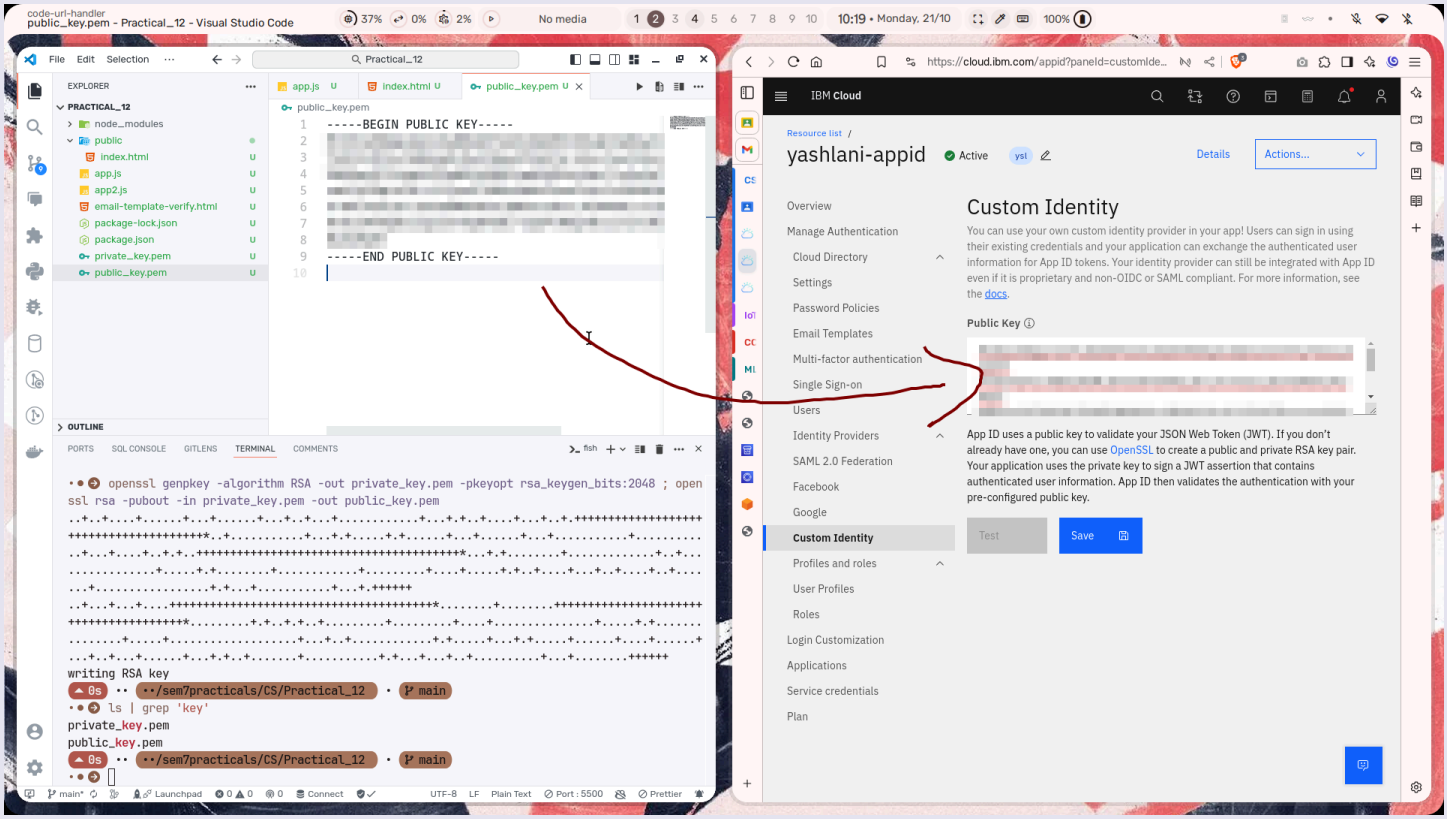


Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 71
CS Practical 12

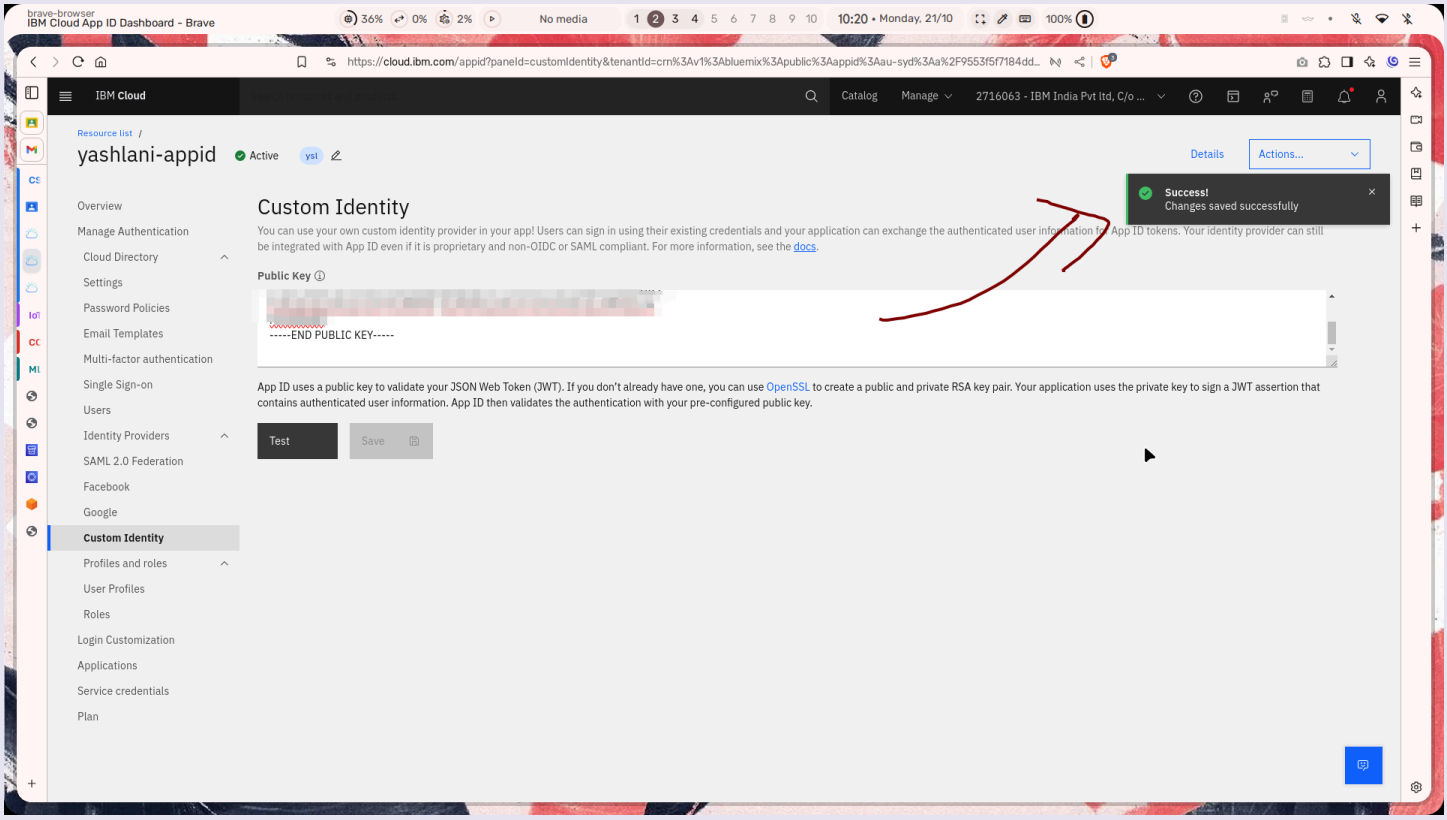


Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 71
CS Practical 12

4. Paste in App ID custom identity provider, the public key generated



Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 71
CS Practical 12



Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 71
CS Practical 12

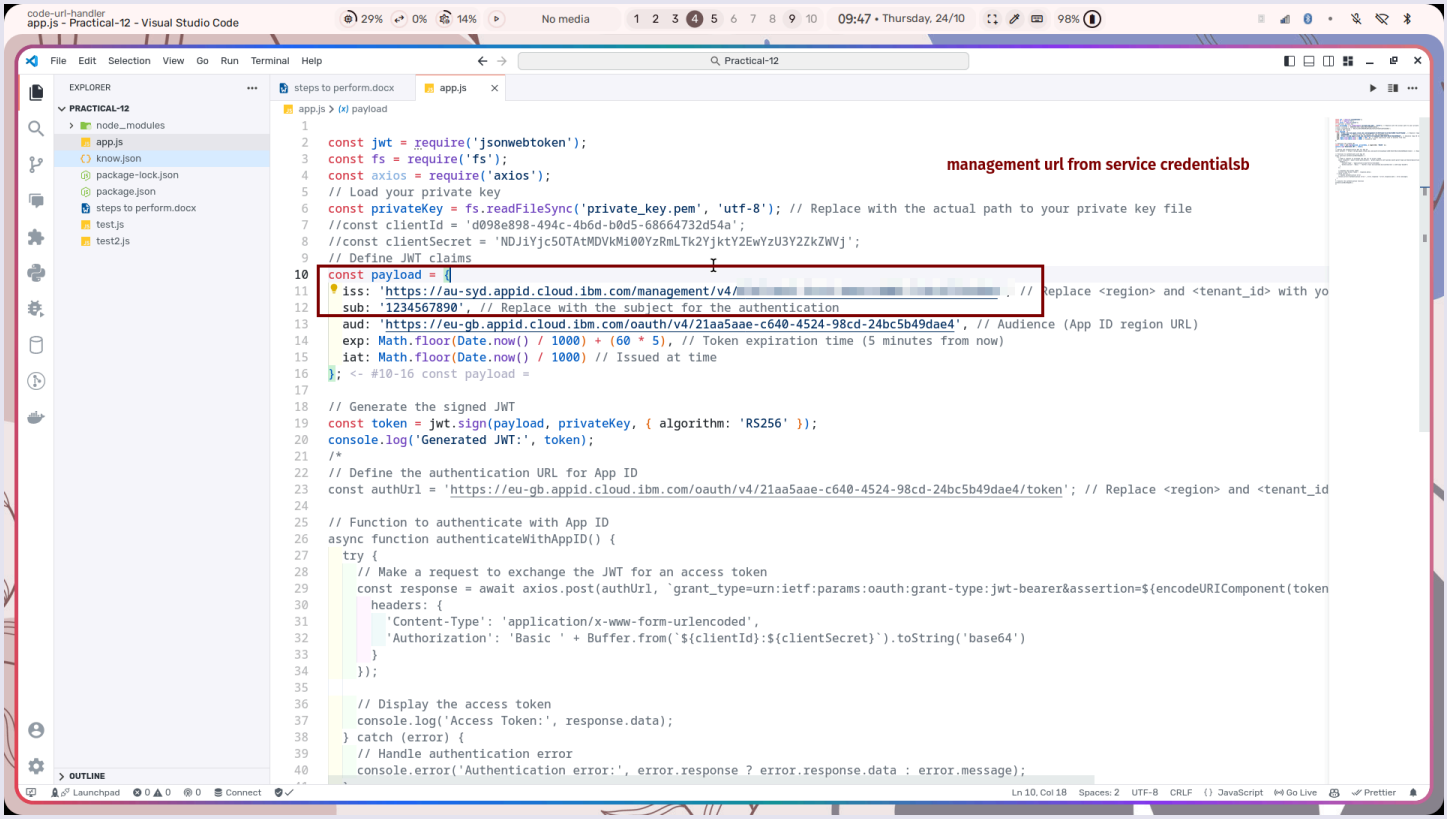
5. Use nodejs app to generate JWT and test it on IBM Cloud

Code :

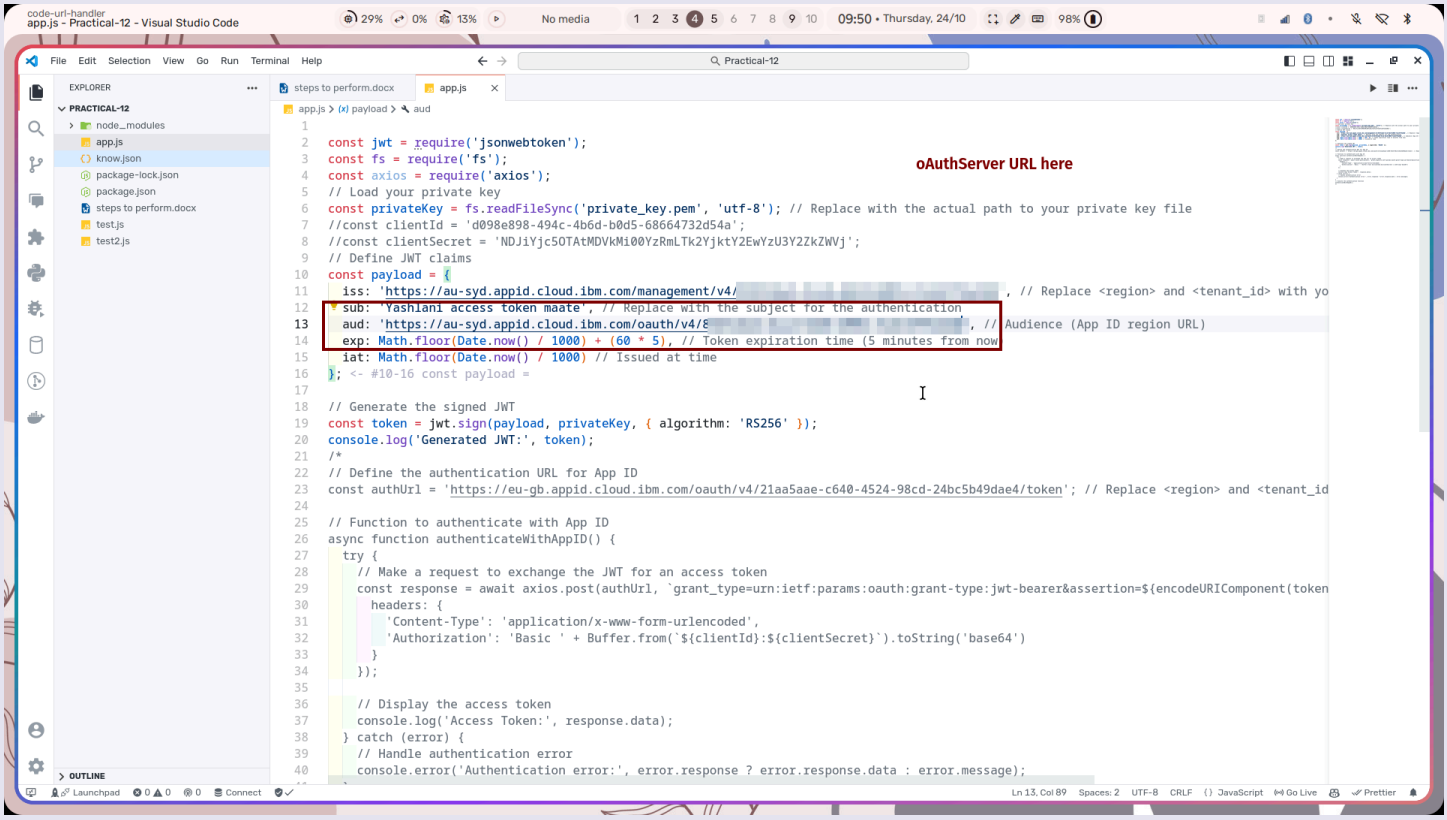
```
const jwt = require('jsonwebtoken');
const fs = require('fs');
// const axios = require('axios');
// Load your private key
const privateKey = fs.readFileSync('private_key.pem', 'utf-8');
// Replace with the actual path to your private key file
//const clientId = 'd098e898-494c-4b6d-b0d5-68664732d54a';
//const clientSecret =
'NDJiYjc5OTAtMDVkMi00YzRmLTk2YjktY2EwYzU3Y2ZkZWVj';
// Define JWT claims
const payload = {
   iss:
'https://au-syd.appid.cloud.ibm.com/management/v4/xxxxxxxxx', //
Replace <region> and <tenant_id> with your App ID region and
tenant ID
   sub: 'Yashlani access token maate', // Replace with the
subject for the authentication
   aud: 'https://au-syd.appid.cloud.ibm.com/oauth/v4/xxxxxxxxx',
// Audience (App ID region URL)
   exp: Math.floor(Date.now() / 1000) + (60 * 2), // Token
expiration time (20 minutes from now)
   iat: Math.floor(Date.now() / 1000) // Issued at time
```

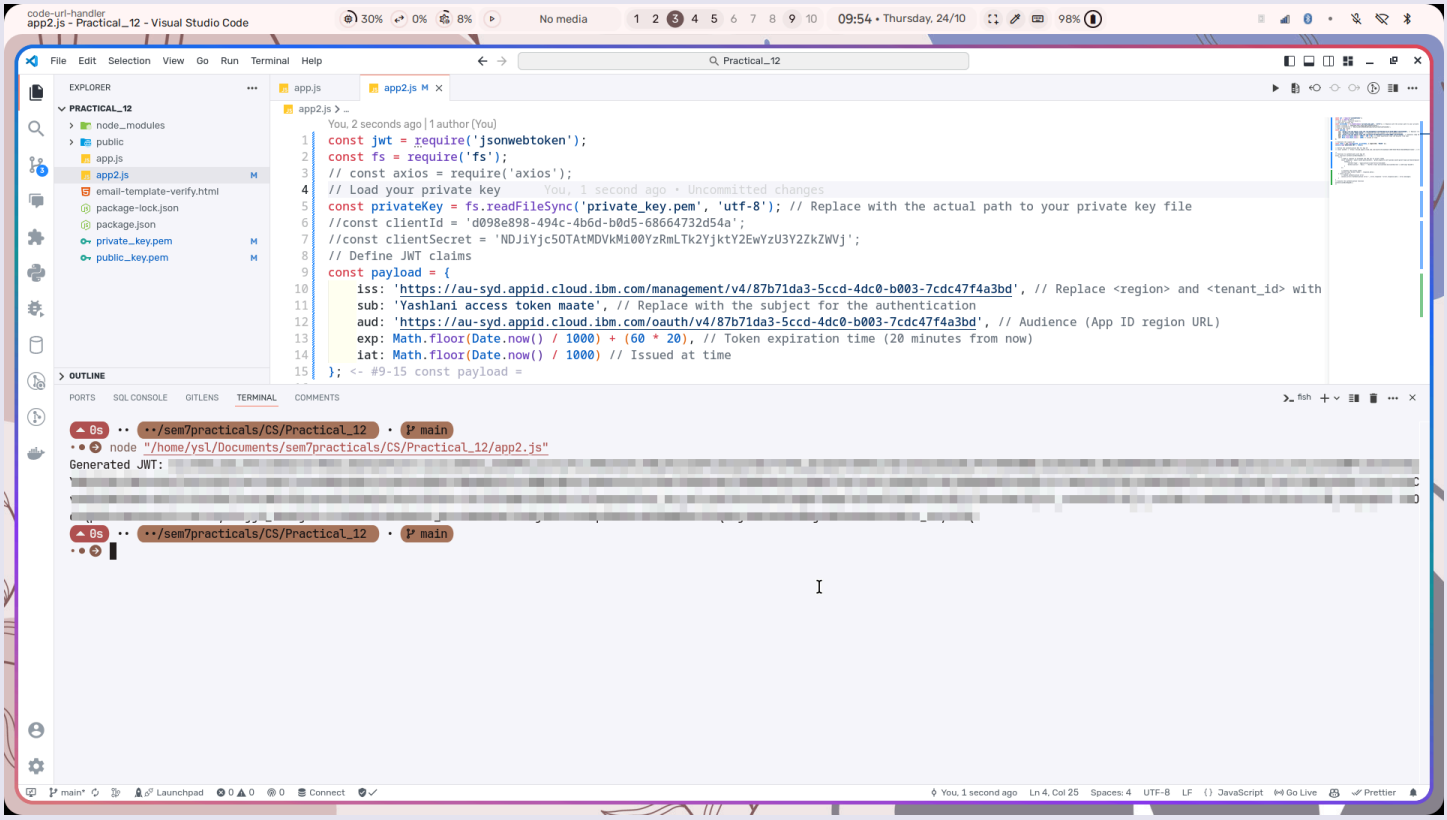
Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 71
CS Practical 12

```
};


// Generate the signed JWT
const token = jwt.sign(payload, privateKey, { algorithm: 'RS256'
});
console.log('Generated JWT:', token);
```

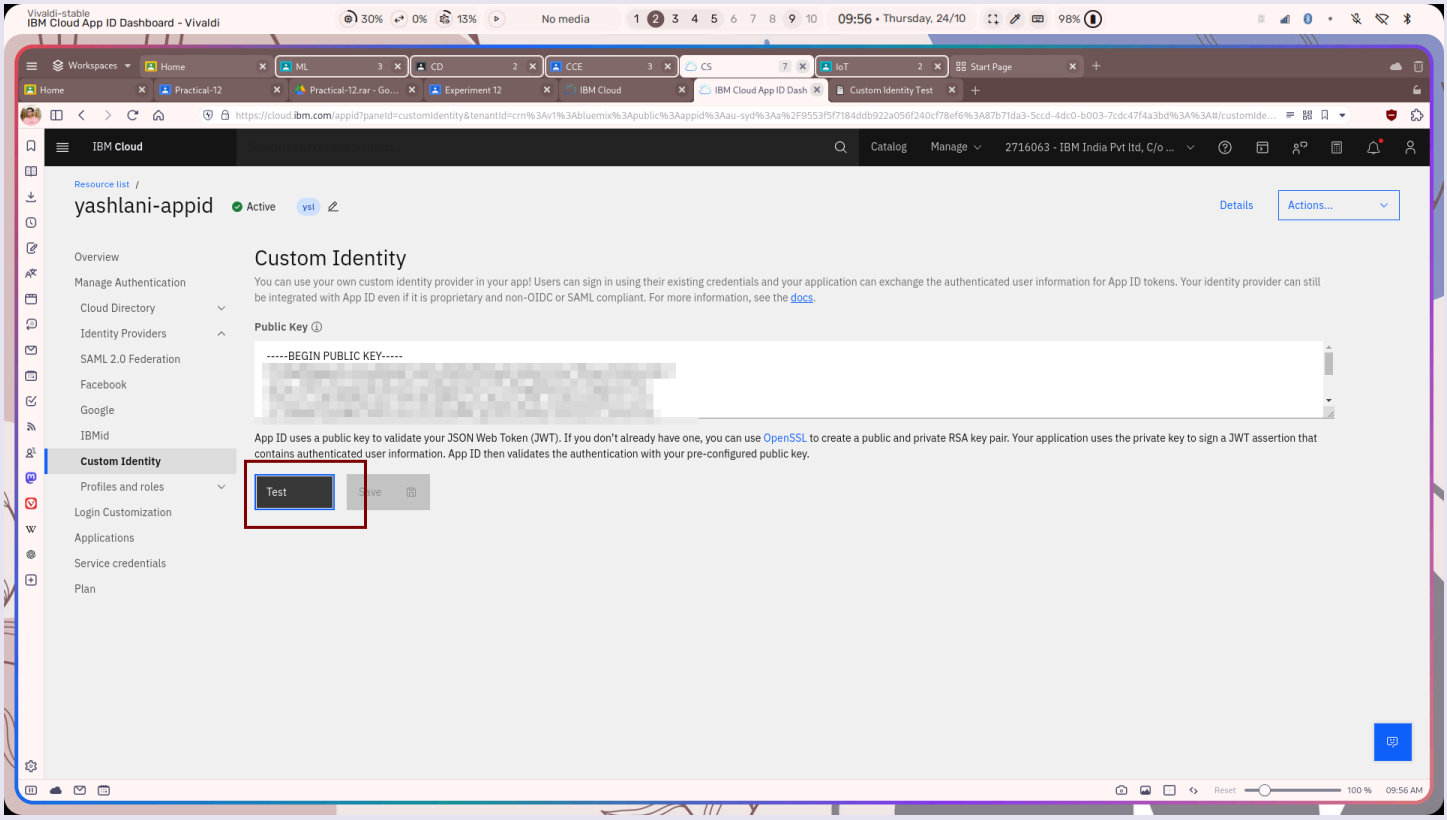


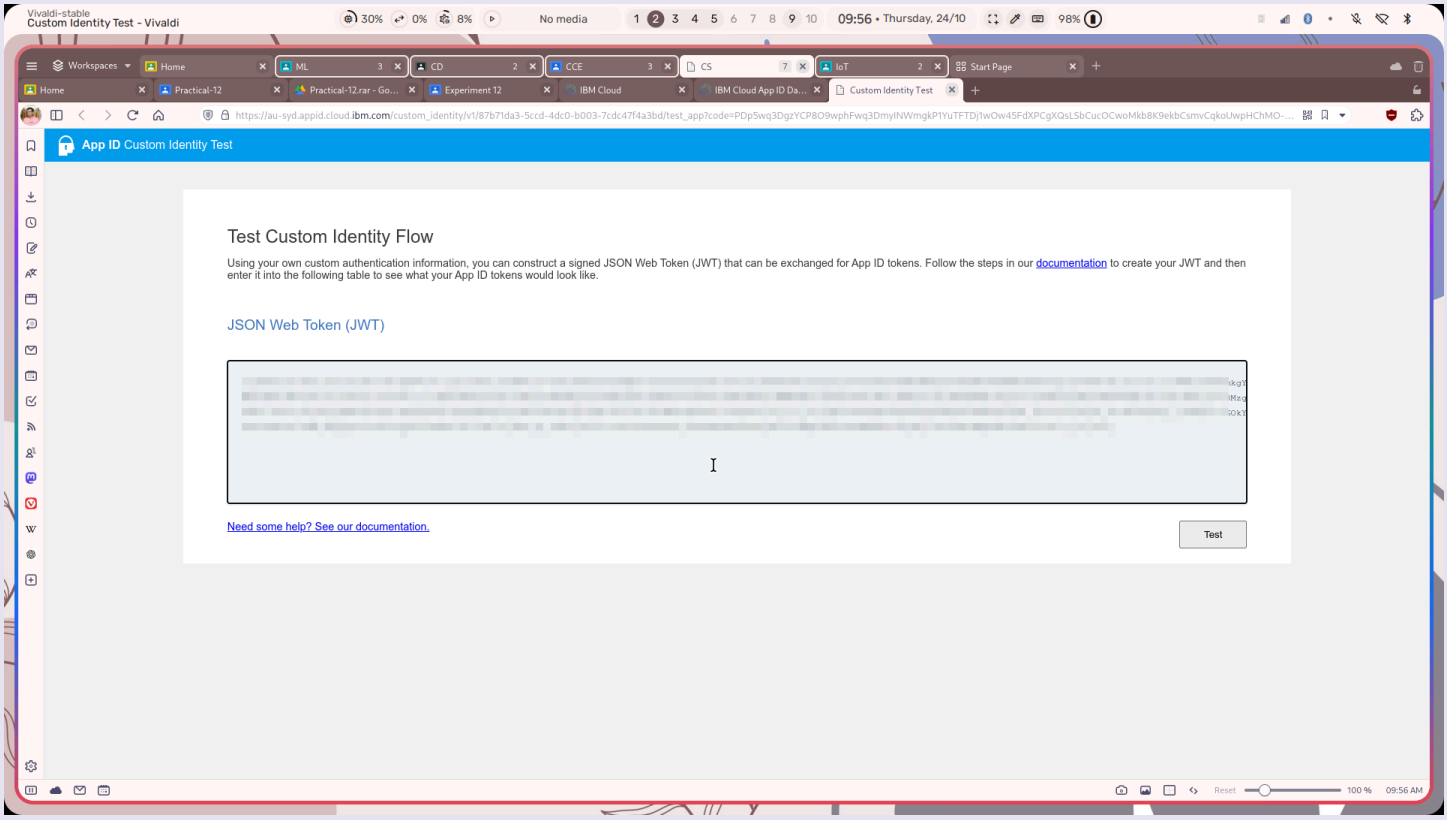
Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 71
CS Practical 12



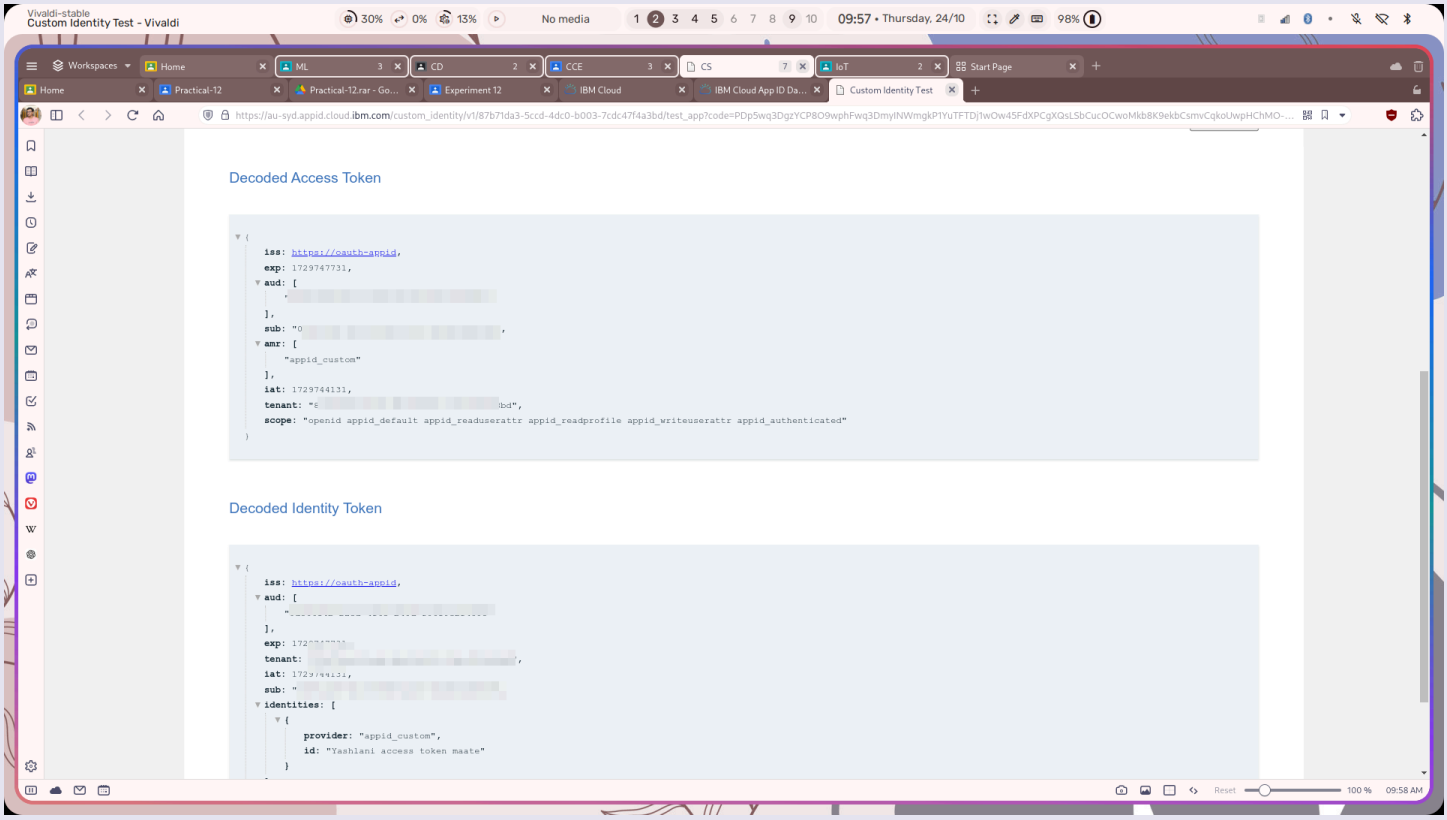


Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 71
CS Practical 12



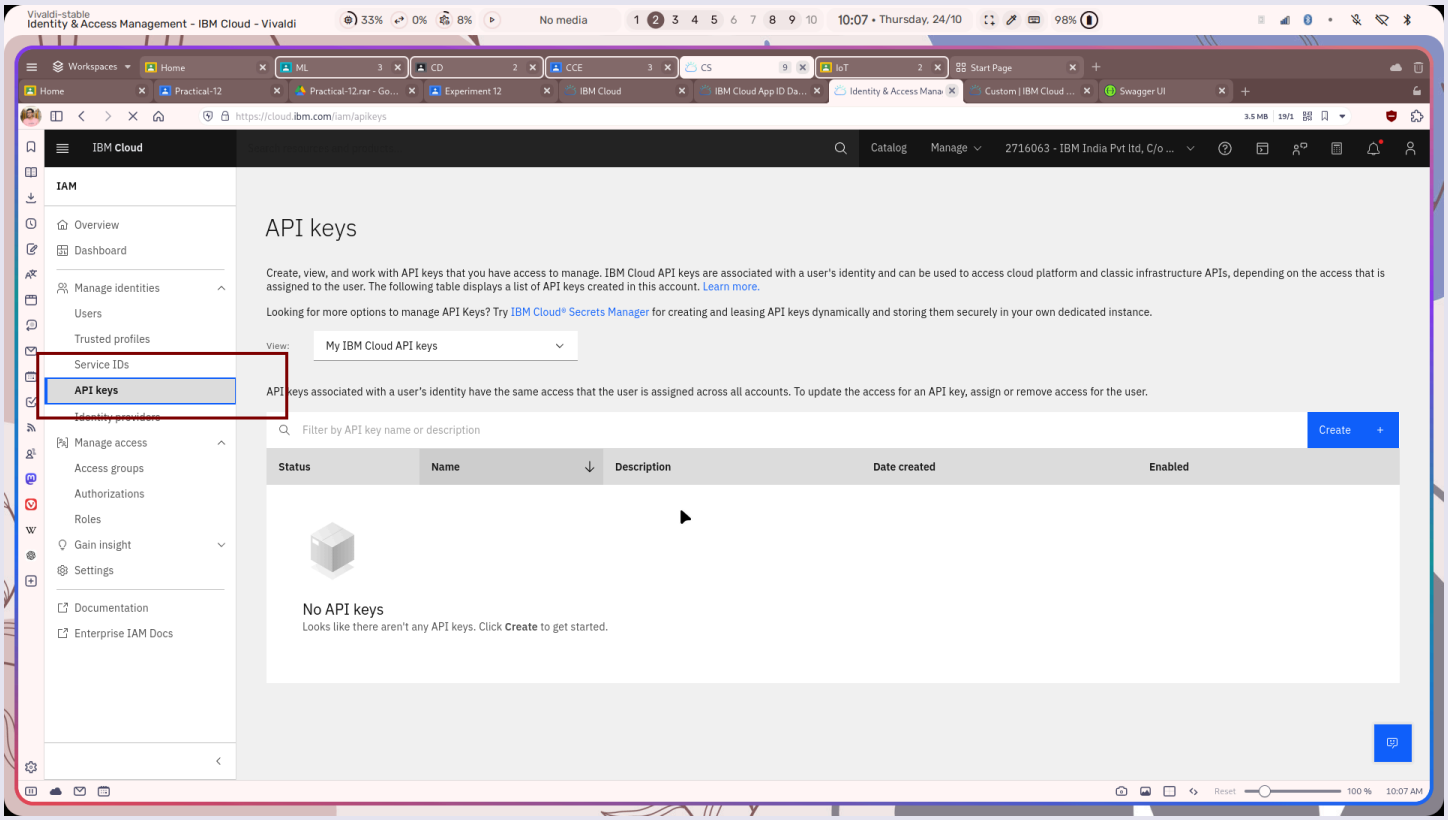


Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 71
CS Practical 12

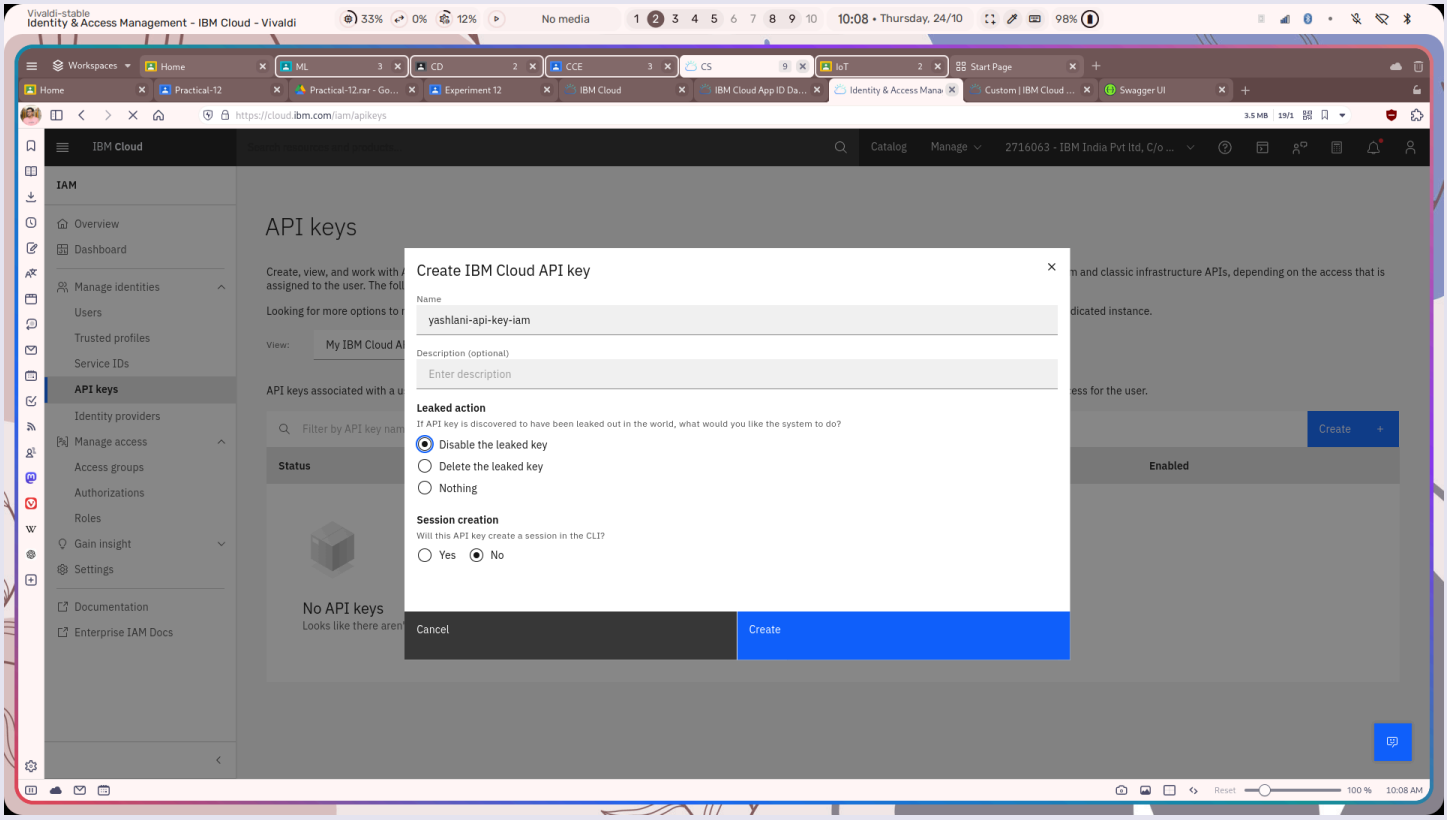


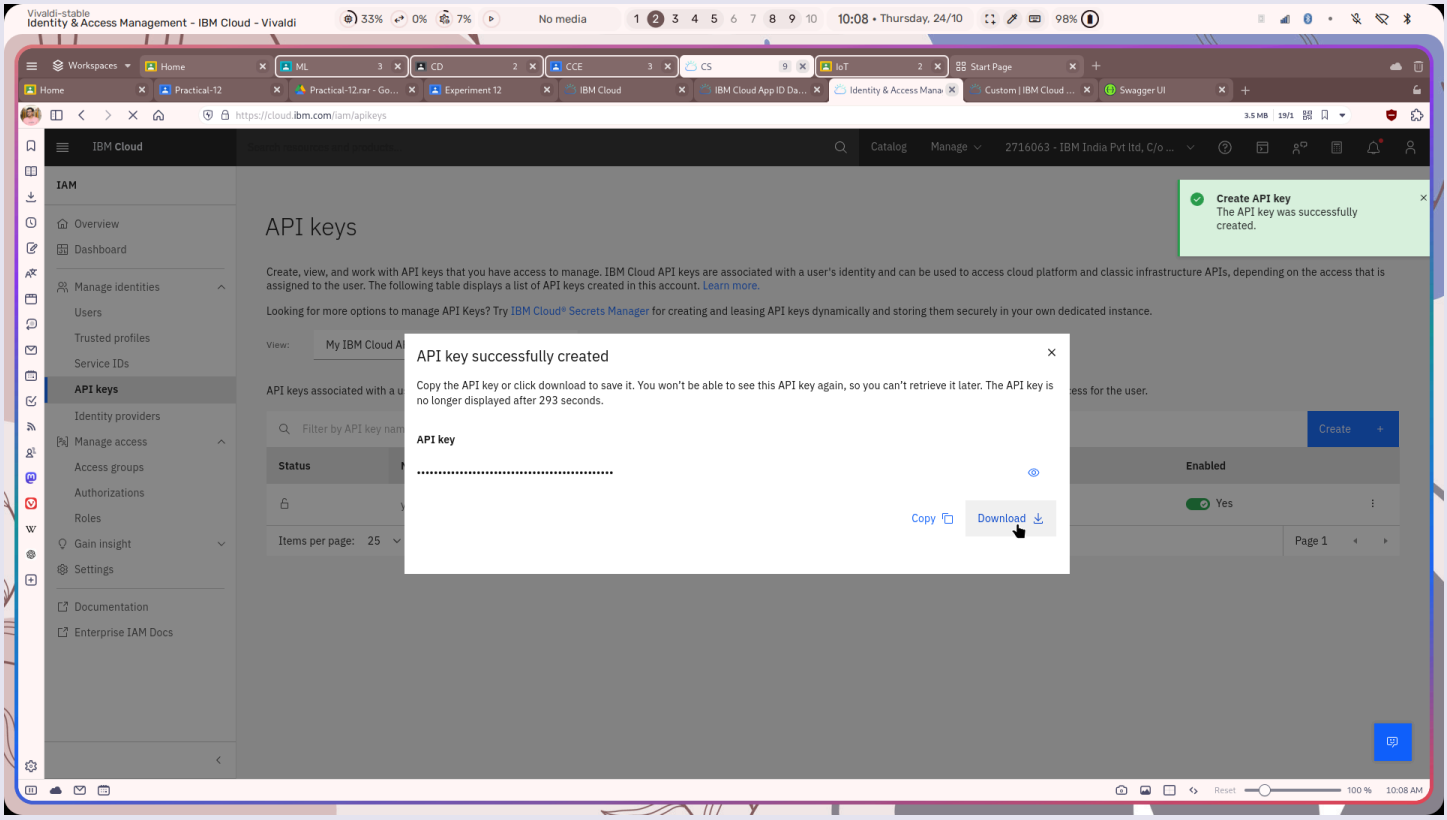
Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 71
CS Practical 12

6. To access public keys using API, authentication is required, for which one time process for getting IAM access token is required



Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 71
CS Practical 12





Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 71
CS Practical 12

7. Use nodejs code to get IAM Access token using API key we generated

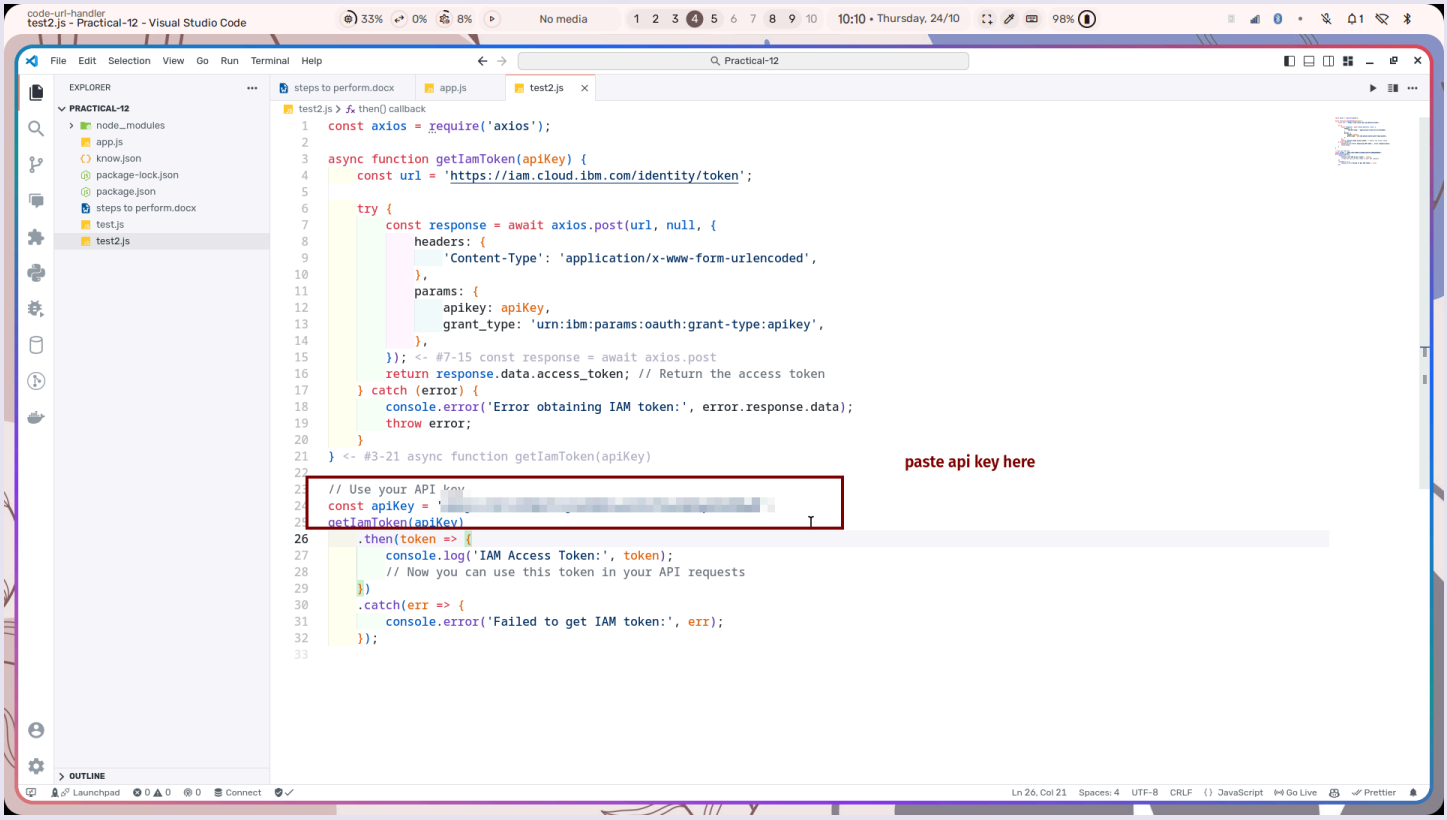
Code :

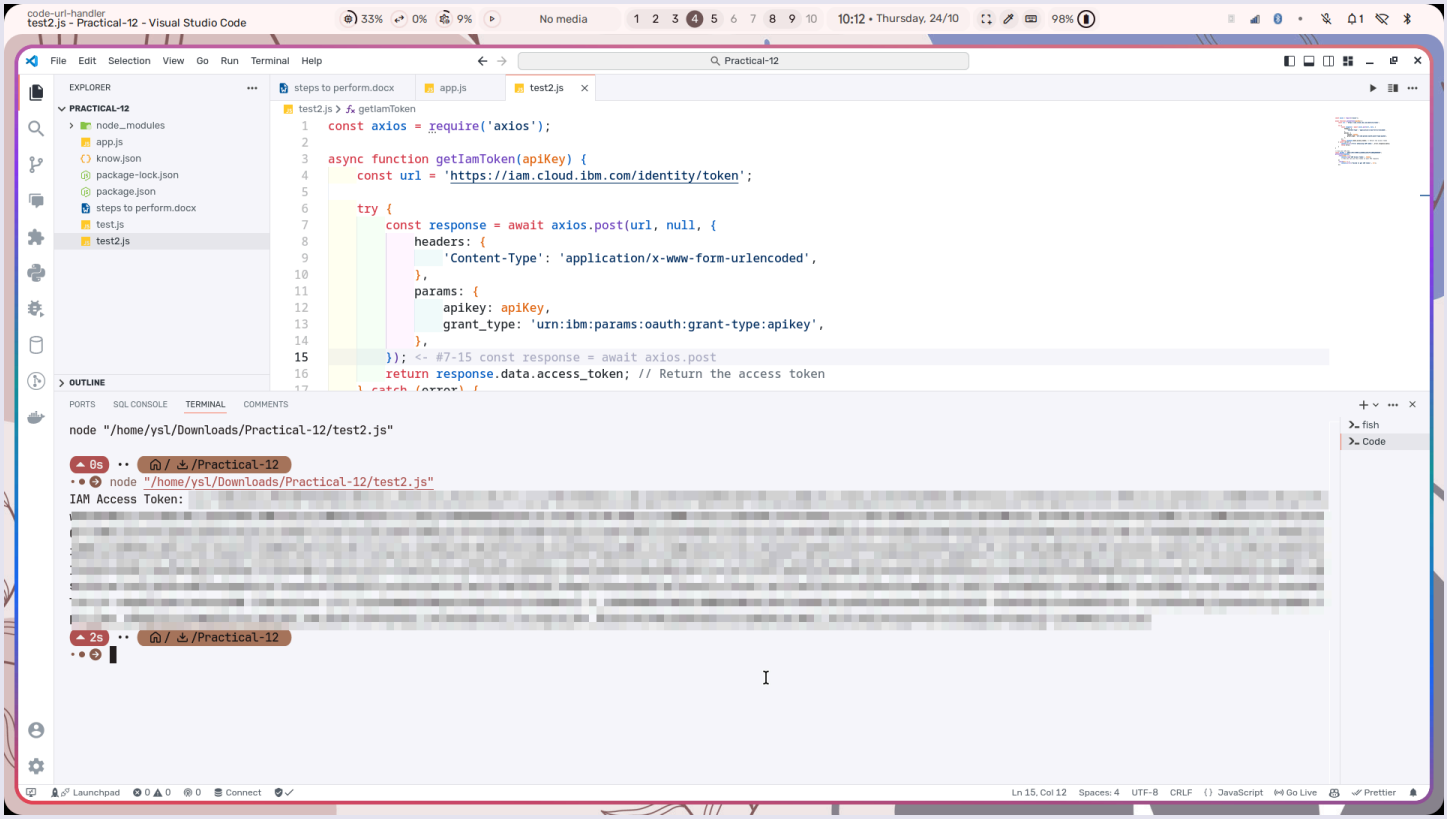
```
const axios = require('axios');

async function getIamToken(apiKey) {
   const url = 'https://iam.cloud.ibm.com/identity/token';

   try {
       const response = await axios.post(url, null, {
           headers: {
               'Content-Type':
'application/x-www-form-urlencoded',
           },
           params: {
               apikey: apiKey,
               grant_type:
'urn:ibm:params:oauth:grant-type:apikey',
           },
       });
       return response.data.access_token; // Return the access
token
   } catch (error) {
       console.error('Error obtaining IAM token:',
error.response.data);
```

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 71
CS Practical 12

```
        throw error;
    }
}

// Use your API key
const apiKey = 'xxxxxxxxxxxxxxxxxxx';
getIamToken(apiKey)
    .then(token => {
        console.log('IAM Access Token:', token);
        // Now you can use this token in your API requests
    })
    .catch(err => {
        console.error('Failed to get IAM token:', err);
    });
```

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 71
CS Practical 12





Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 71
CS Practical 12

When API is used to get public key from IBM Cloud App ID using JWT and then verifying it, authentication is required otherwise anyone can access our public key. Hence for this purpose, this IAM Access Token will be used

8. Now, use nodejs another code to test access to resources using this IAM access token got as output

Code :

```
const express = require('express');
const jwt = require('jsonwebtoken');
const axios = require('axios');

const app = express();
const port = 3000;

// Your App ID public key URL for verifying the JWT signature
const APP_ID_PUBLIC_KEY_URL =
'https://au-syd.appid.cloud.ibm.com/management/v4/xxxxxxxxxxxxxx
xxxxxxxxxxx/config/idps/custom';

// Middleware to verify the JWT
async function verifyToken(req, res, next) {
   const authHeader = req.headers['authorization'];

   if (!authHeader || !authHeader.startsWith('Bearer ')) {
       return res.status(401).json({ error: 'Authorization
```

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 71
CS Practical 12

```
header missing or malformed' });
    }

    const token = authHeader.split(' ')[1];

    try {
        // Fetch the App ID public keys
        const response = await axios.get(APP_ID_PUBLIC_KEY_URL, {
            headers: {
                Authorization: `Bearer xxxxxxxxxxxx` // Pass the
Bearer token here
            },
        });
        console.log(response)
        const publicKeys = response.data.config.publicKey;
        console.log(publicKeys);
        if (!publicKeys || publicKeys.length === 0) {
            return res.status(500).json({ error: 'Failed to
retrieve public keys' });
        }

        // Decode the JWT header to find the key ID (kid)
        const decodedHeader = jwt.decode(token, { complete: true
});
```

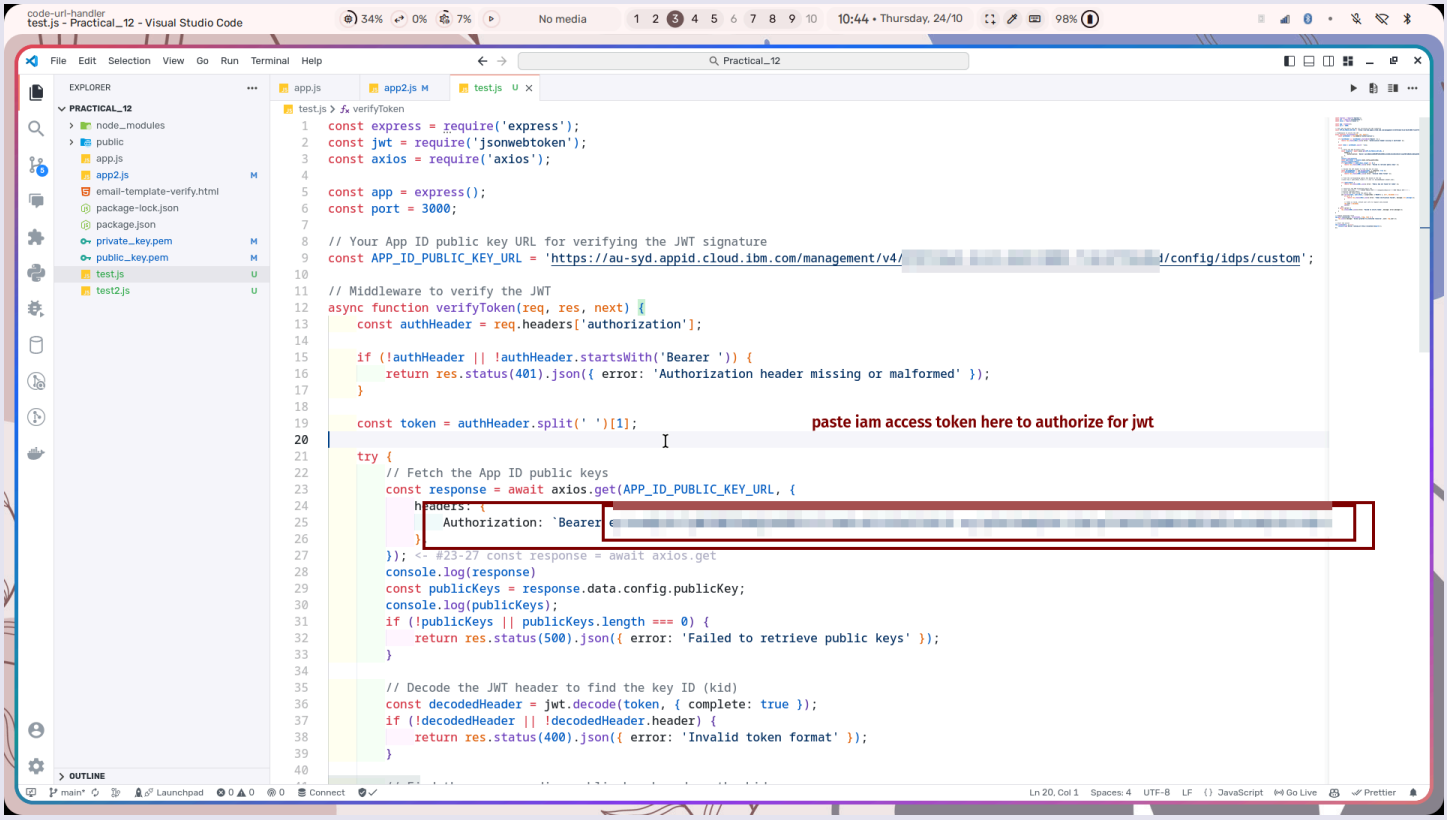
Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 71
CS Practical 12

```
        if (!decodedHeader || !decodedHeader.header) {
            return res.status(400).json({ error: 'Invalid token
format' });
        }

        // Find the corresponding public key based on the kid
        //const key = publicKeys.find(k => k.kid ===
decodedHeader.header.kid);

        if (!publicKeys) {
            return res.status(400).json({ error: 'Public key not
found for token' });
        }

        // Construct the PEM formatted public key
        // const publicKey = `-----BEGIN PUBLIC
KEY-----\n${publicKeys}\n-----END PUBLIC KEY-----`;
        //console.log(publicKey);
        // Verify the token using the public key
        jwt.verify(token, publicKeys, { algorithms: ['RS256'] },
(err, decoded) => {
            if (err) {
                return res.status(401).json({ error: 'Token
verification failed', message: err.message });
```

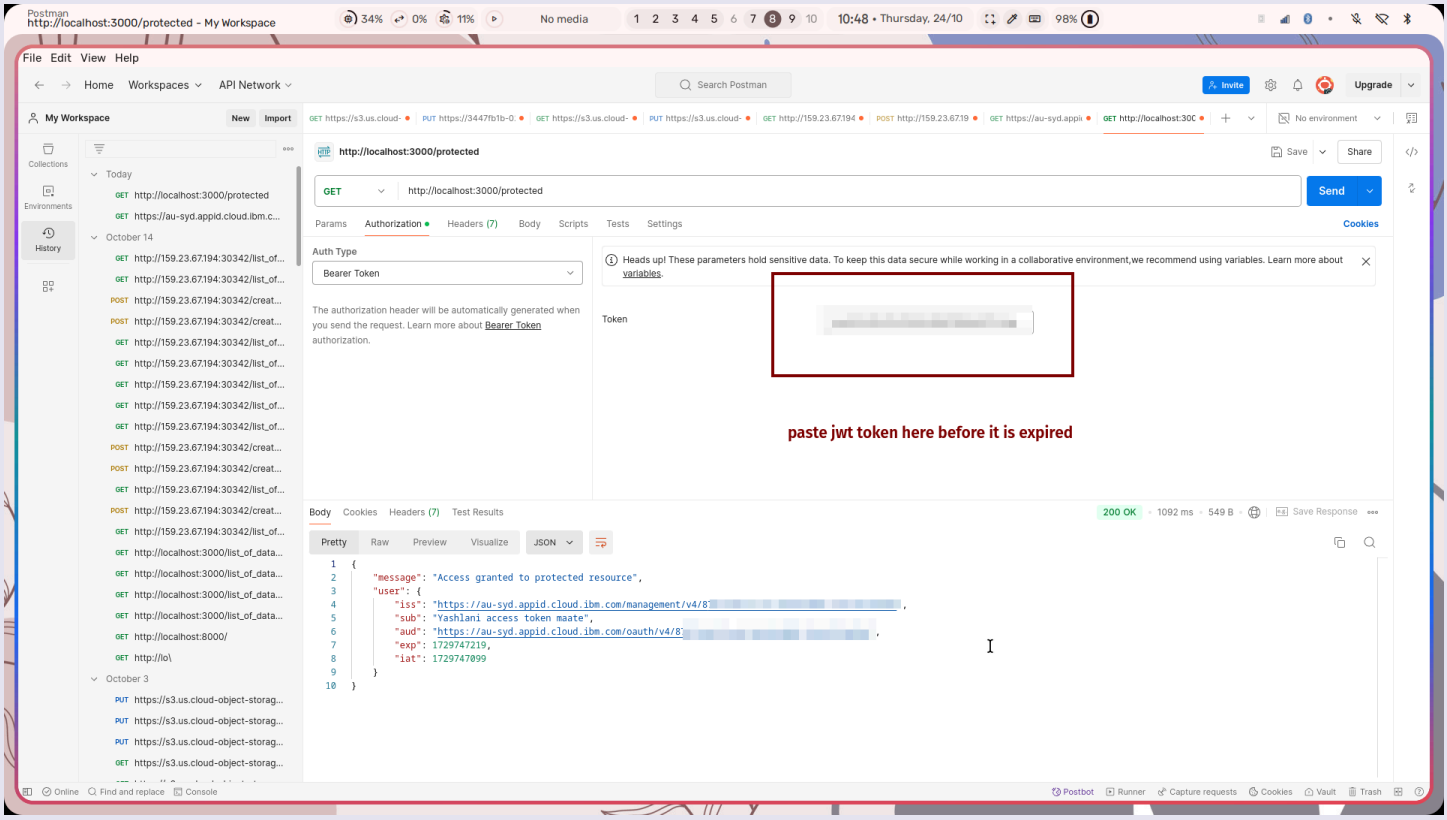
Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 71
CS Practical 12

```
            }

            // Token is valid, attach user info to request and
proceed
            req.user = decoded;
            next();
        });
    } catch (error) {
        res.status(500).json({ error: 'Failed to verify token',
message: error.message });
    }
}

// Sample protected route
app.get('/protected', verifyToken, (req, res) => {
    res.json({ message: 'Access granted to protected resource',
user: req.user });
});

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

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 71
CS Practical 12

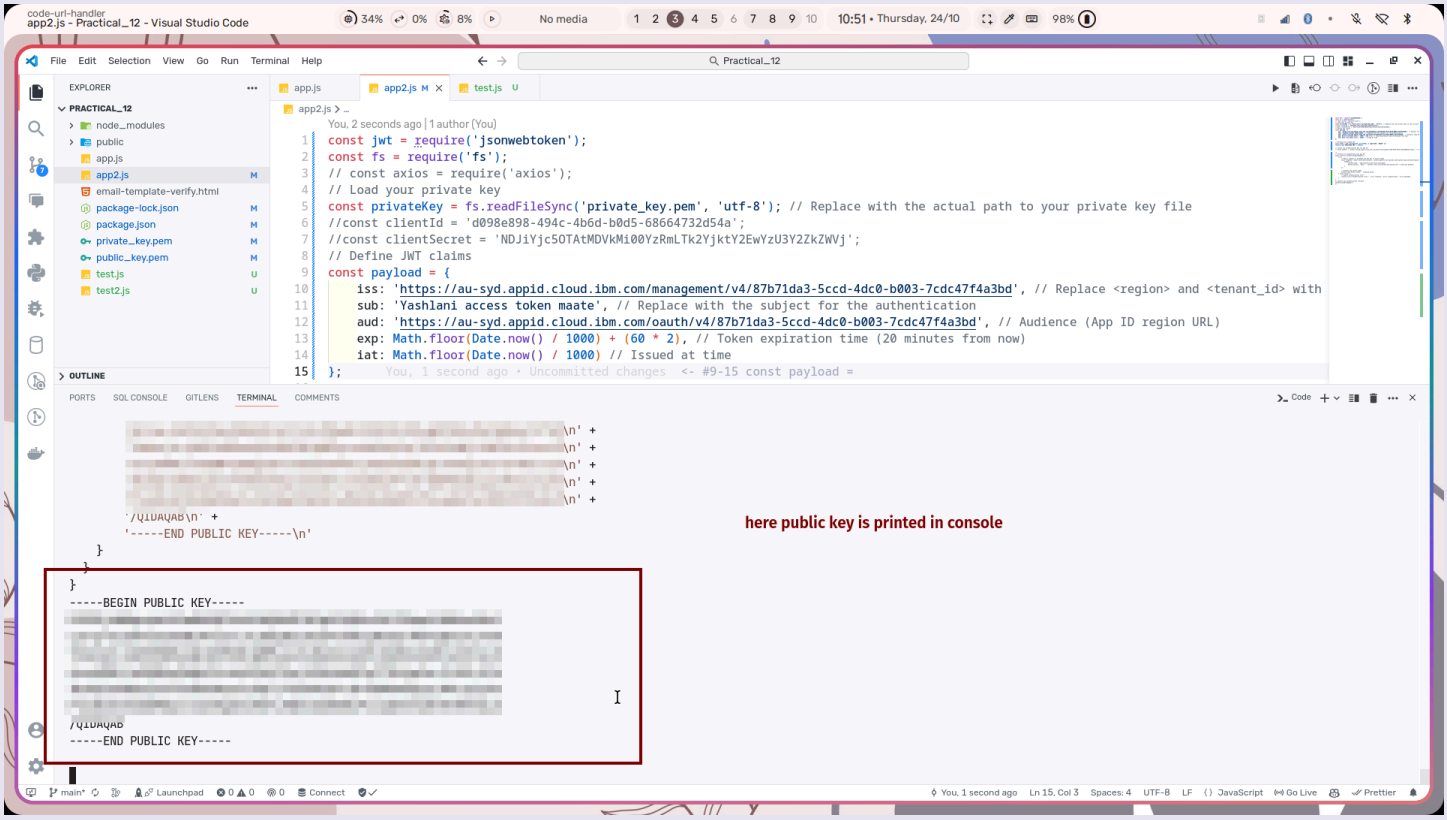


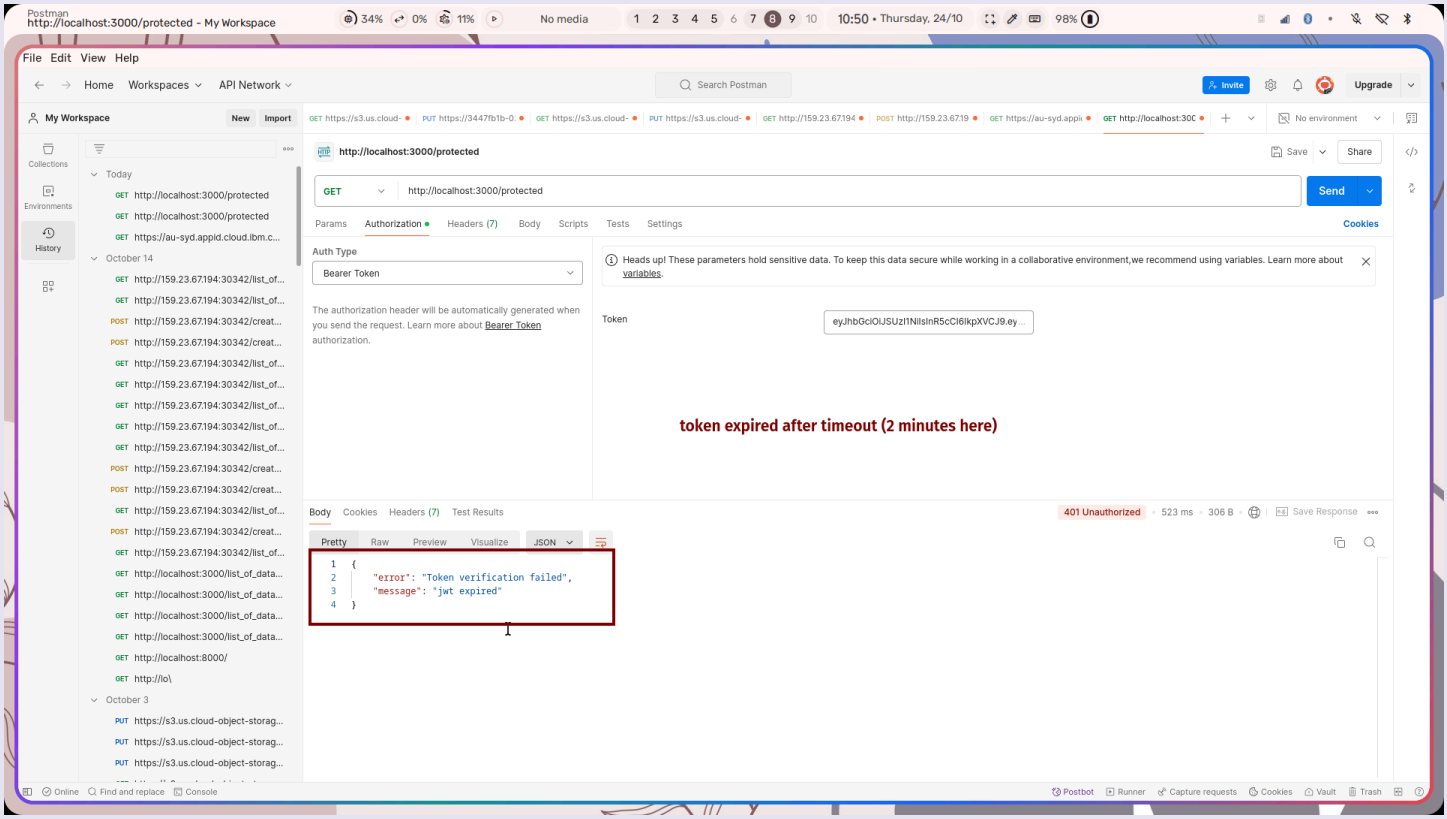
Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 71
CS Practical 12



In this code, the public key in App ID is verified using JWT as authentication but IAM access token is also used for the purpose of accessing public key in authorized manner from cloud

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 71
CS Practical 12





Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 71
CS Practical 12

Conclusion :