Institute of Computer Technology

B. Tech Computer Science and Engineering

Sub: Identity and Access Management (2CSE507)

Name - Nisarg Prajapati

Branch - Cyber Security

Enrollment no. – 23162171019

Semester - 5

Class - A

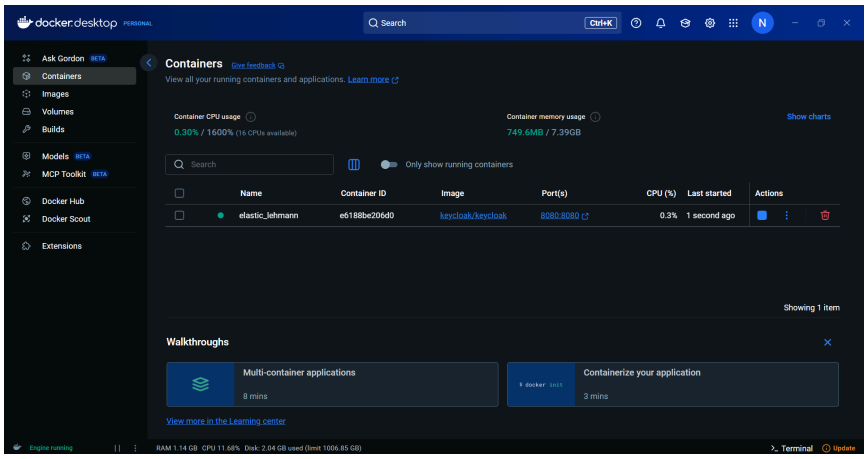
Batch – 52

# PRACTICAL NO:- 7

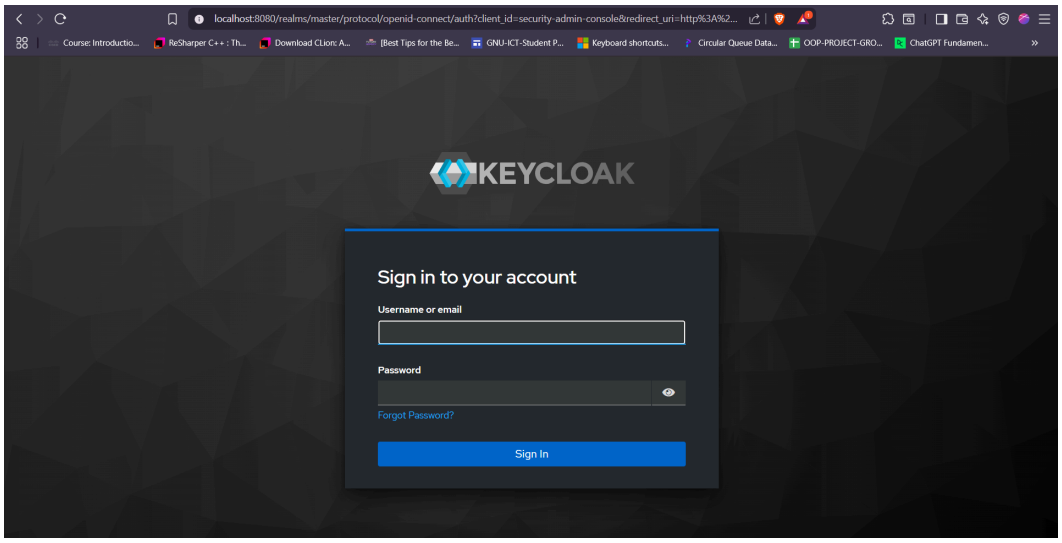
**Aim :** Demonstration of SSO in Keycloak

## Starting the Container:

- Open your Docker Desktop.
- In the containers tab, click the start button of the container you want to start.



- Now open your browser and go to ‘http://localhost:8080/’.



## Code:

- App1:

  ❖ index.js

```
import express from "express";
import session from "express-session";
import { client, code_verifier, code_challenge } from
"./auth.js";

const PORT = 3001;
const app = express();

app.use(
  session({
    secret: "app1_secret",
    resave: false,
    saveUninitialized: true,
  })
);

// Home route
app.get("/", async (req, res) => {
  if (!req.session.tokenSet) {
    return res.send(`<a href="/login">Login with
Keycloak</a>`);
  }
  res.send(`
    <h2>Welcome to App1!</h2>
    <p>User: ${req.session.user?.preferred_username}</p>
    <a href="/logout">Logout</a>
    <br><a href="http://localhost:3002/">Go to App2</a>
  `);
});

// Login route
app.get("/login", (req, res) => {
  req.session.code_verifier = code_verifier;
```

```
  const authUrl = client.authorizationUrl({
    scope: "openid profile email",
    code_challenge,
    code_challenge_method: "S256",
  });
  res.redirect(authUrl);
});

// Callback route
app.get("/callback", async (req, res) => {
  try {
    const params = client.callbackParams(req);
    const tokenSet = await client.callback(
      "http://localhost:3001/callback",
      params,
      { code_verifier: req.session.code_verifier }
    );
    req.session.tokenSet = tokenSet;

    const userinfo = await
client.userinfo(tokenSet.access_token);
    req.session.user = userinfo;

    res.redirect("/");
  } catch (err) {
    console.error(err);
    res.status(500).send("Login failed");
  }
});

// Logout route
app.get("/logout", (req, res) => {
  const idToken = req.session.tokenSet?.id_token;
  req.session.destroy(() => {
    let logoutUrl =
`http://localhost:8080/realms/demo-realm/protocol/openid-conne
ct/logout`;
    if (idToken) {
      logoutUrl +=
`?id_token_hint=${idToken}&post_logout_redirect_uri=http://loc
```

```
alhost:3001/`;
    }
    res.redirect(logoutUrl);
  });
});

app.listen(PORT, () =>
  console.log(` App1 running on http://localhost:${PORT}`)
);
```

- auth.js

```
import { createClient } from "../oidc_config/oidc-config.js";

export const { client, code_verifier, code_challenge } =
createClient(
  "app-a",
  "http://localhost:3001/callback"
);
```

- App2

  - ❖ index.js

```
import express from "express";
import session from "express-session";
import { client, code_verifier, code_challenge } from
"./auth.js";
import axios from "axios";

const PORT = 3002;
const app = express();

app.use(
  session({
    secret: "app2_secret",
    resave: false,
    saveUninitialized: true,
  })
);

// Home route
app.get("/", async (req, res) => {
  if (!req.session.tokenSet) {
    // Attempt SSO: check if user is logged in Keycloak using
iframe-like request
    return res.send(`<a href="/login">Login with
Keycloak</a>`);
  }
  res.send(`
    <h2>Welcome to App2!</h2>
    <p>User: ${req.session.user?.preferred_username}</p>
    <a href="/logout">Logout</a>
    <br><a href="http://localhost:3001/">Go to App1</a>
  `);
});

// Login route
app.get("/login", (req, res) => {
```

```
  req.session.code_verifier = code_verifier;
  const authUrl = client.authorizationUrl({
    scope: "openid profile email",
    code_challenge,
    code_challenge_method: "S256",
  });
  res.redirect(authUrl);
});

// Callback route
app.get("/callback", async (req, res) => {
  try {
    const params = client.callbackParams(req);
    const tokenSet = await client.callback(
      "http://localhost:3002/callback",
      params,
      { code_verifier: req.session.code_verifier }
    );
    req.session.tokenSet = tokenSet;

    const userinfo = await
client.userinfo(tokenSet.access_token);
    req.session.user = userinfo;

    res.redirect("/");
  } catch (err) {
    console.error(err);
    res.status(500).send("Login failed");
  }
});

// Logout route
app.get("/logout", (req, res) => {
  const idToken = req.session.tokenSet?.id_token;
  req.session.destroy(() => {
    let logoutUrl =
`http://localhost:8080/realms/demo-realm/protocol/openid-conne
ct/logout`;
    if (idToken) {
      logoutUrl +=
```

```
`?id_token_hint=${idToken}&post_logout_redirect_uri=http://loc
alhost:3002/`;
    }
    res.redirect(logoutUrl);
  });
});

app.listen(PORT, () =>
  console.log(` App2 running on http://localhost:${PORT}`)
);
```

❖ auth.js

```
import { createClient } from "../oidc_config/oidc-config.js";

export const { client, code_verifier, code_challenge } =
createClient(
  "app-b",
  "http://localhost:3002/callback"
);
```

- my-realm.json

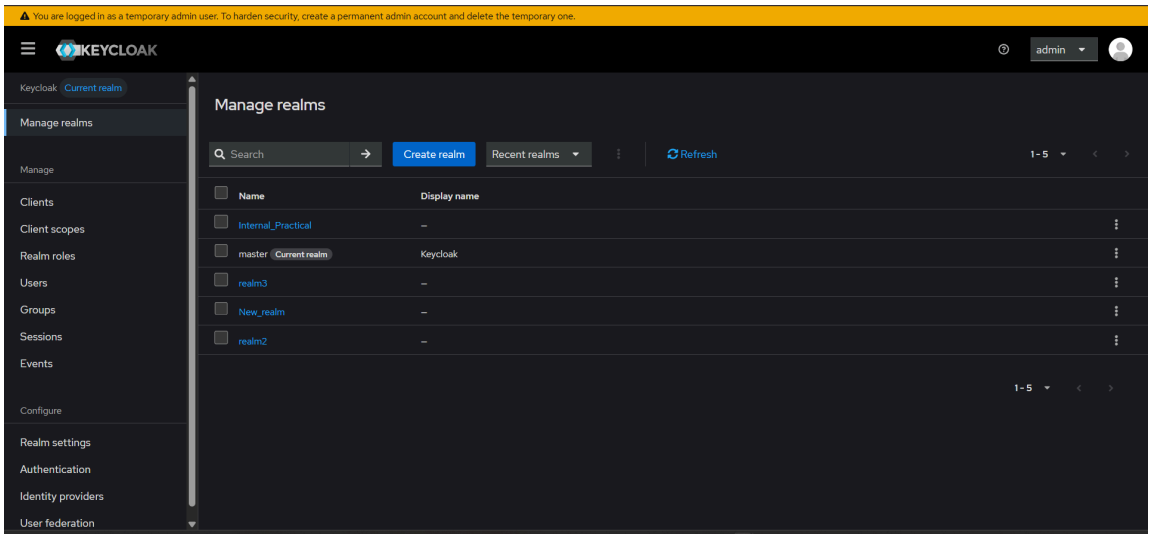
```
{
  "realm": "demo-realm",
  "enabled": true,
  "clients": [
    {
      "clientId": "app-a",
      "publicClient": true,
      "protocol": "openid-connect",
      "redirectUris": ["http://localhost:3000/callback"],
      "webOrigins": ["http://localhost:3000"],
      "standardFlowEnabled": true,
      "implicitFlowEnabled": false,
      "directAccessGrantsEnabled": true
```

```
    },
    {
      "clientId": "app-b",
      "publicClient": true,
      "protocol": "openid-connect",
      "redirectUris": ["http://localhost:3001/callback"],
      "webOrigins": ["http://localhost:3001"],
      "standardFlowEnabled": true,
      "implicitFlowEnabled": false,
      "directAccessGrantsEnabled": true
    }
  ],
  "users": [
    {
      "username": "testuser",
      "enabled": true,
      "emailVerified": true,
      "firstName": "Test",
      "lastName": "User",
      "email": "testuser@example.com",
      "credentials": [
        {
          "type": "password",
          "value": "Test@123",
          "temporary": false
        }
      ]
    }
  ]
}
```

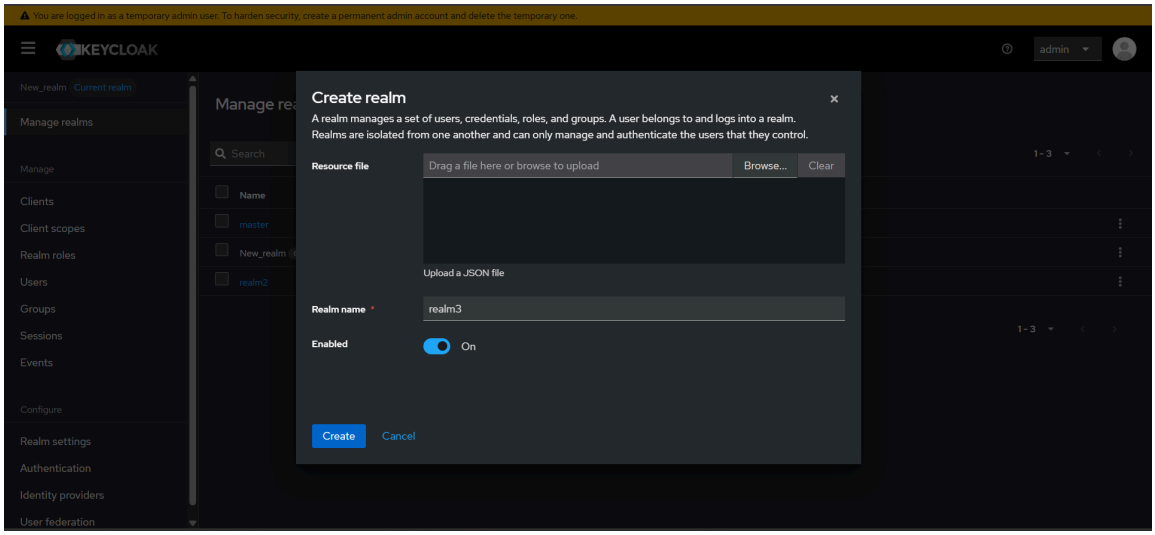
# Creating a new Realm:

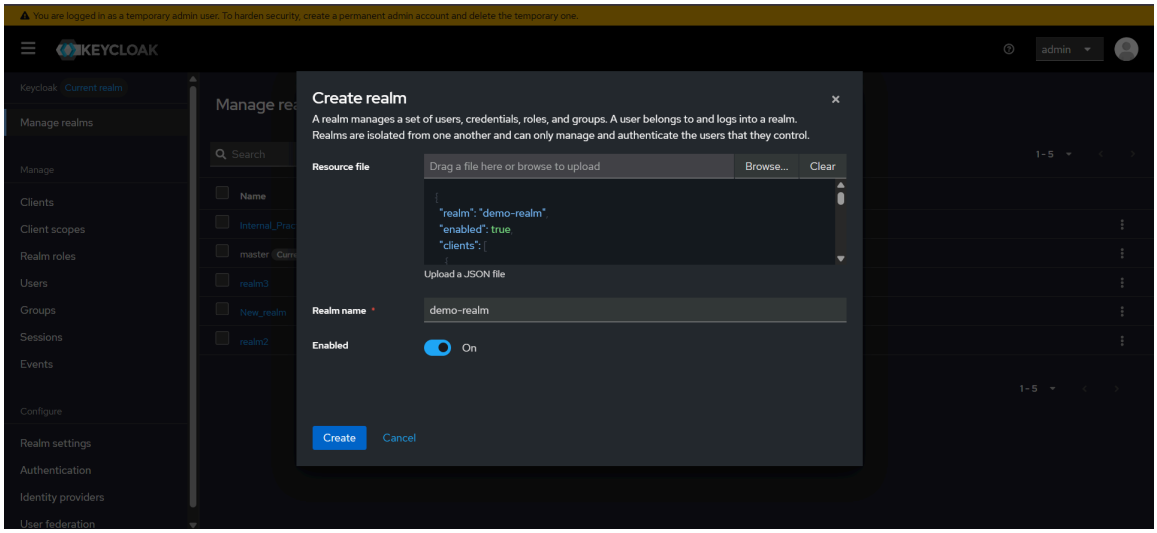
- Login as admin.


- Go to the ‘Manage Realm’ tab and click on ‘Create realm’.

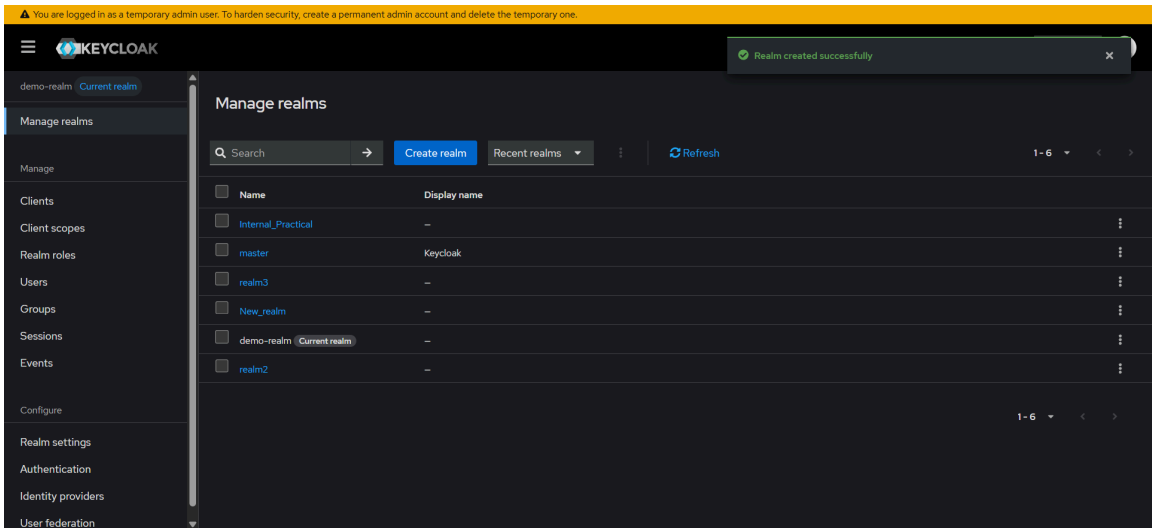


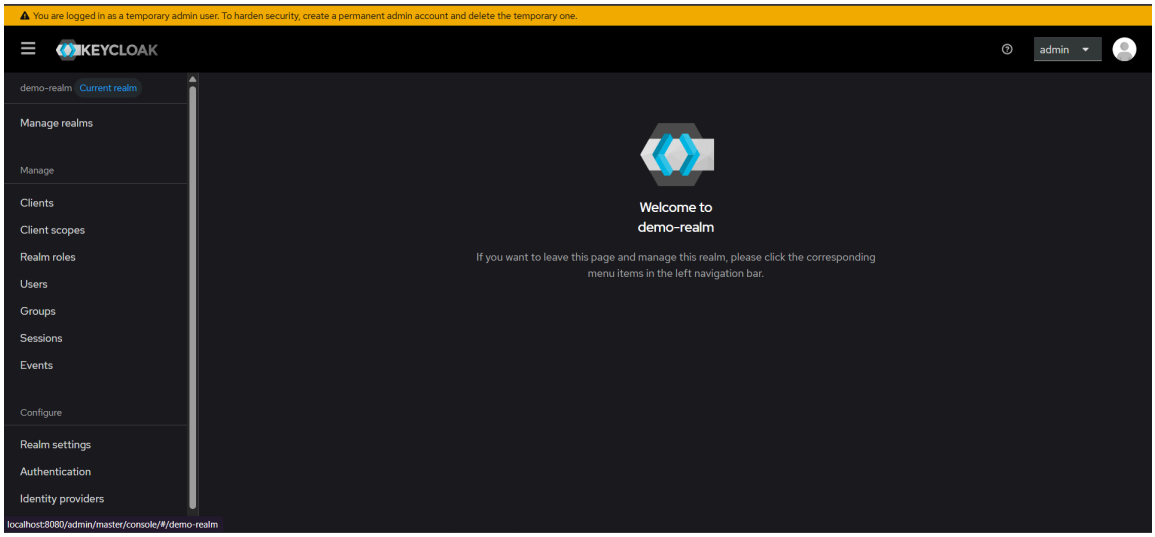
- Enter the realm name, select the ‘my-realm.json’ in ‘Resource file’ and create the realm.



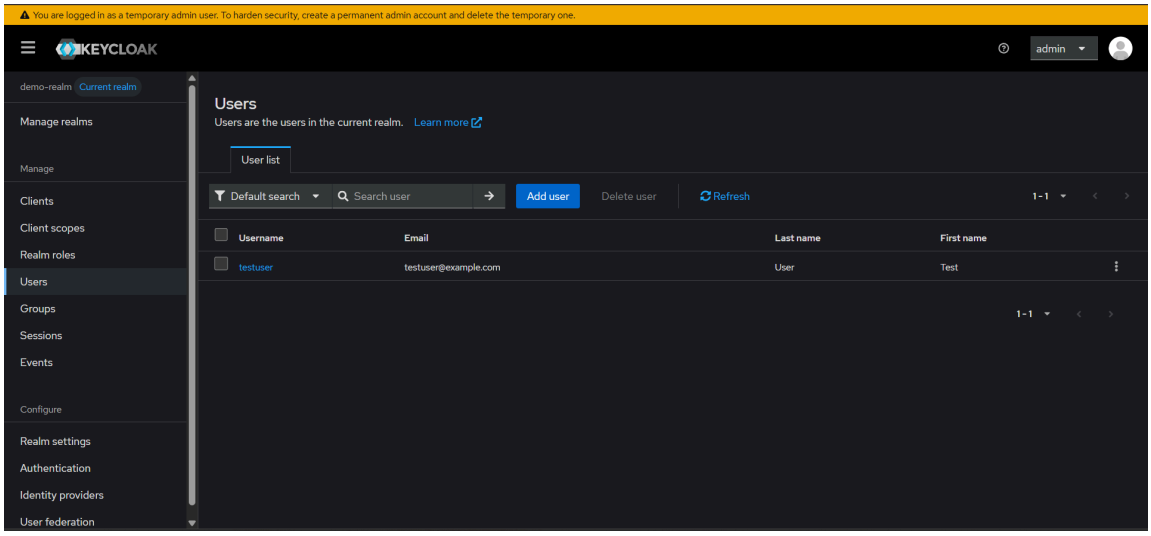



- To change between realms, in ‘Manage Realm’ tab, click on the realm name.

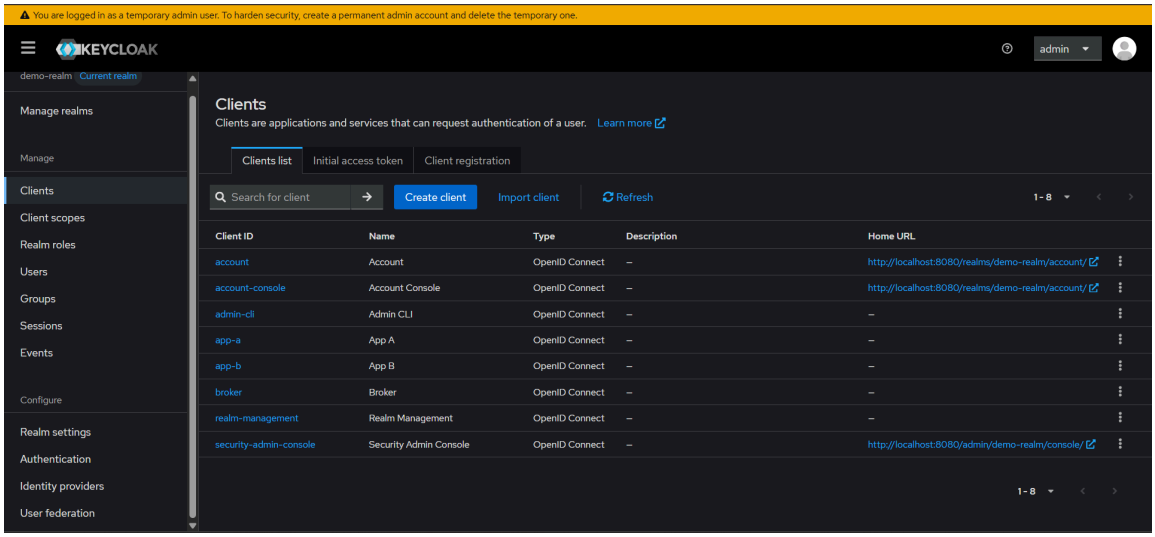


- Check the created user in ‘Users’ tab

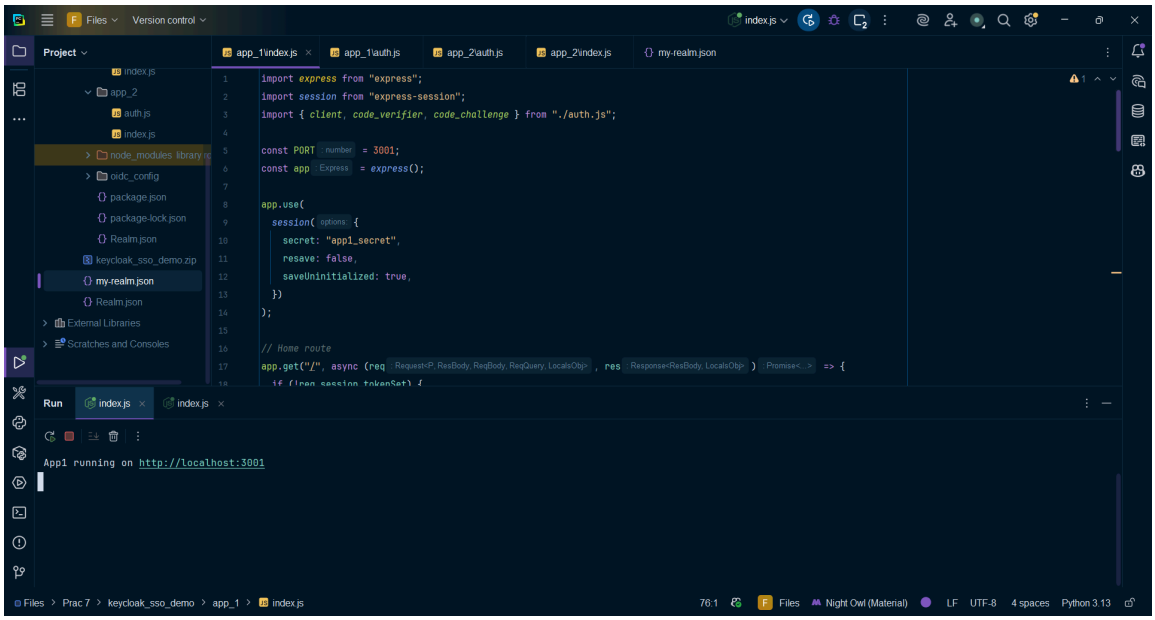


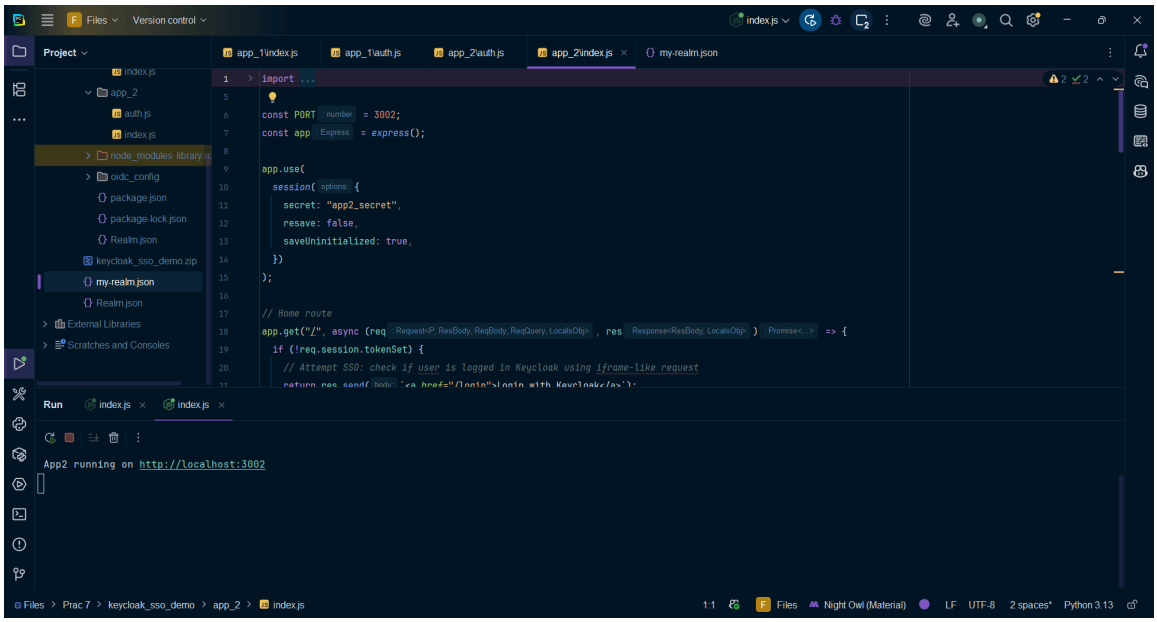

- Check the created clients in ‘Clients’ tab



## Starting the login apps:

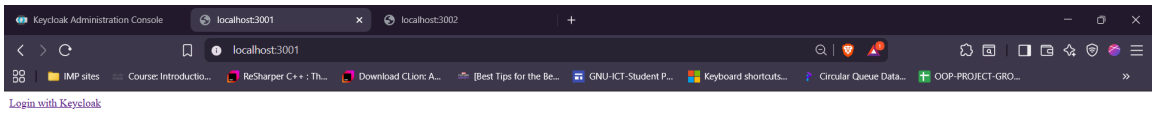
- Run the index.js files from both app1 and app2.



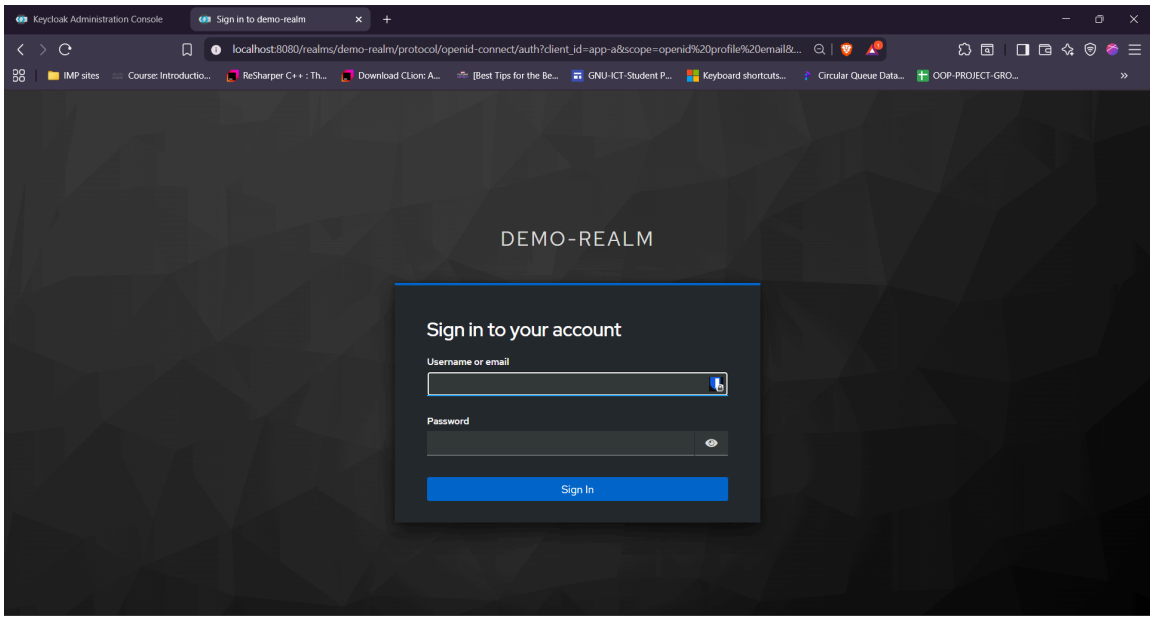
# Checking SSO:

- Open ‘http://localhost:3001/’ in the browser.

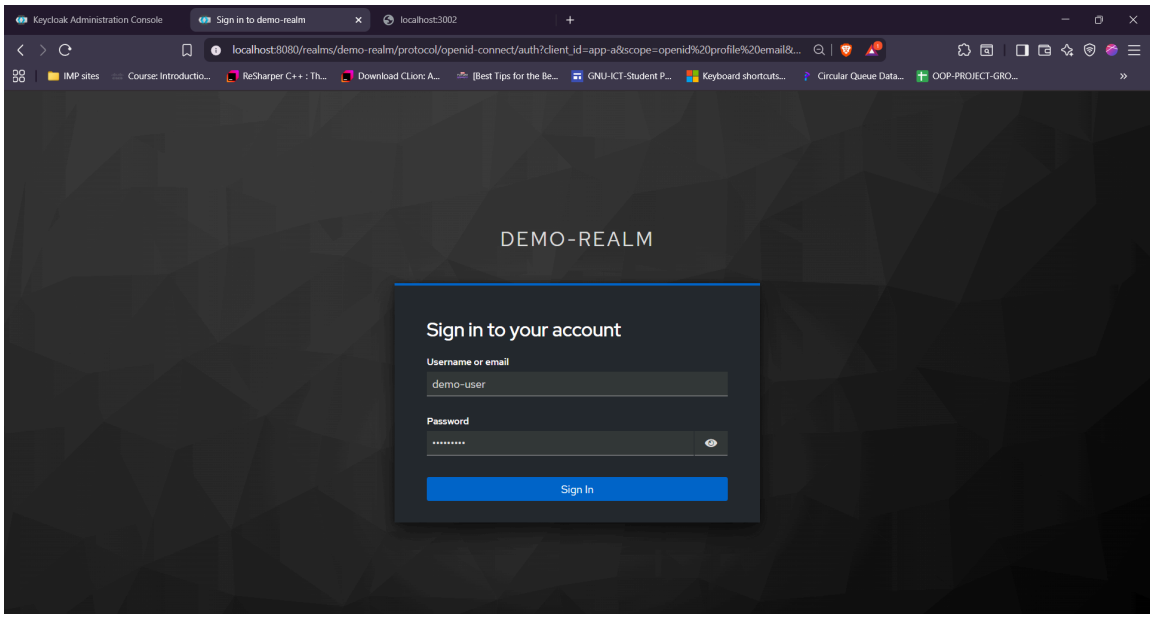


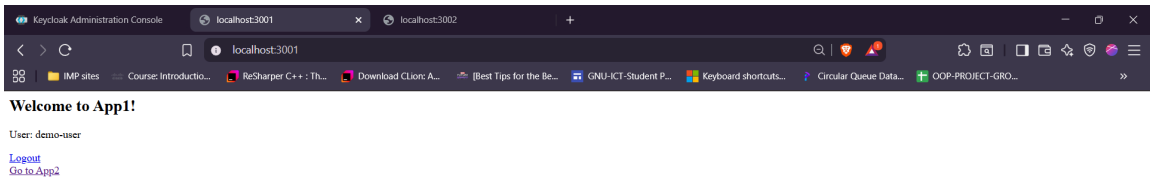

---

- Click on ‘Login with Keyclock’.

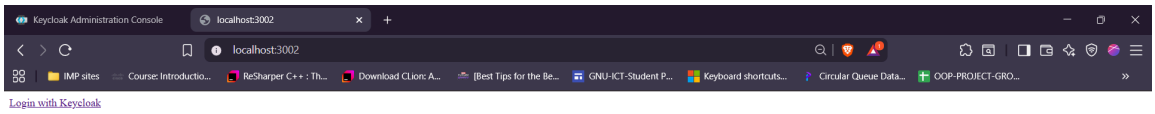


- Enter username and password and click on ‘Sign in’.



- Click on ‘Go to App2’.




- Click on ‘Login with Keyclock’.