A4 – OpenID authentication

1. Introduction

The goal for this assignment is to implement a web application and deploy it on the virtual machine we were given at the start of the course. The web application should be use Microsoft's EntraID OpenID to authenticate and authorize users. This means that the resources should be unavailable for unauthorized users.

2. Background

2.1. JSON web tokens (JWT)

JSON web tokens are a standard for secure information transmission, commonly used for authorization. The tokens are made up of three main components: header, payload and signature.

The header typically contains information about the token, and the algorithm used for creating signature. The payload contains the information to be transmitted. The signature contains both the header and payload base64url encoded along with a secret which is signed by following the algorithm defined in the header. This ensures that the token is unaltered (jwt.io, 2024).

3. Design and implementation

3.1. Authentication

To implement a web application that integrates with Microsoft's EntaID, the object *identity.web.Auth* was used. This supports functions like *log_in(), log_out* and *get_user(),* which makes the authentication process easy.

When a user navigates to the web page, they are redirected to the login page, where they have an option to start the authentication process by clicking "sign in". This will call the log_in() function. The user is not able to reach any other endpoints until they are authorized.

3.2 Sending requests

After a user is successfully logged in, the server receives an access token from Microsoft's authentication system. This token is what the server uses to make requests, for example for accessing additional information about the user, or updating existing information.

3.3 OpenID scopes

There are three OpenID scopes that are used for this implementation. Firstly, the User.Read scope is used for signing in, and read user profile. User.ReadBasic.All makes the users' basic profiles available and is necessary for using the List users API (List Users API, 2024). User.WriteRead gives access to reading and writing to a user profile and is necessary for using the Update user API (Update User API, 2024).

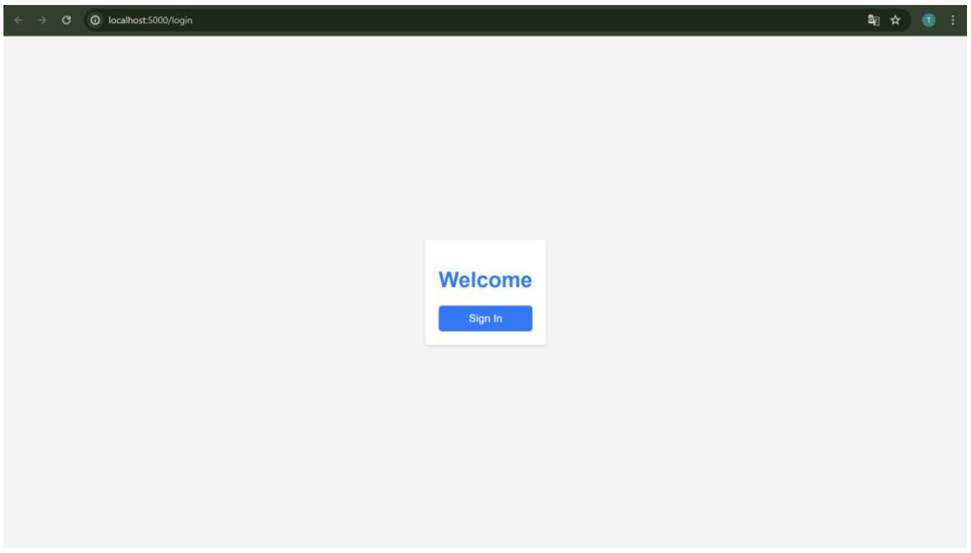
4 Evaluation

4.2 Web application functionality

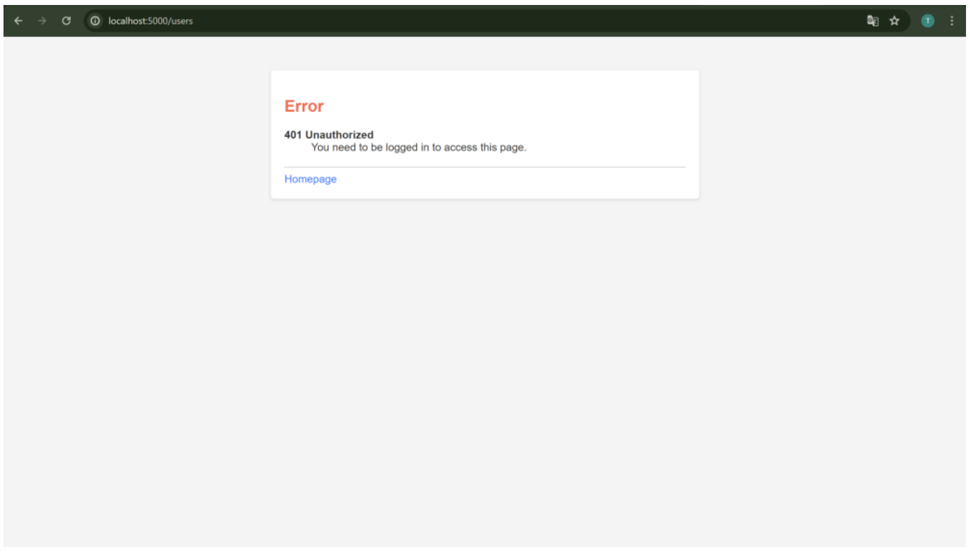
To run the web application on localhost, the following command can be used (Se README.md for full instructions:

```
flask run --debug --host=localhost --port=5000
```

When visiting http://localhost:5000 after running this command, the following page shows up:



If a user tries to access a forbidden endpoint before signing in, this will not be successful:

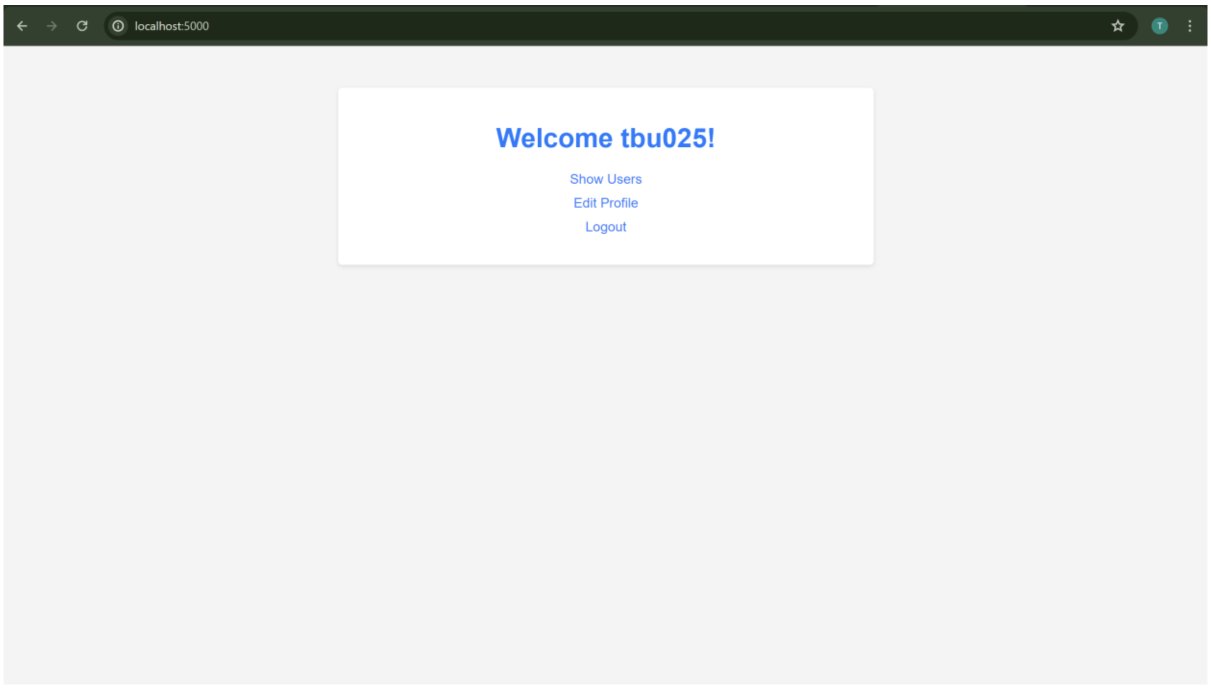


Error

401 Unauthorized
You need to be logged in to access this page.

Homepage

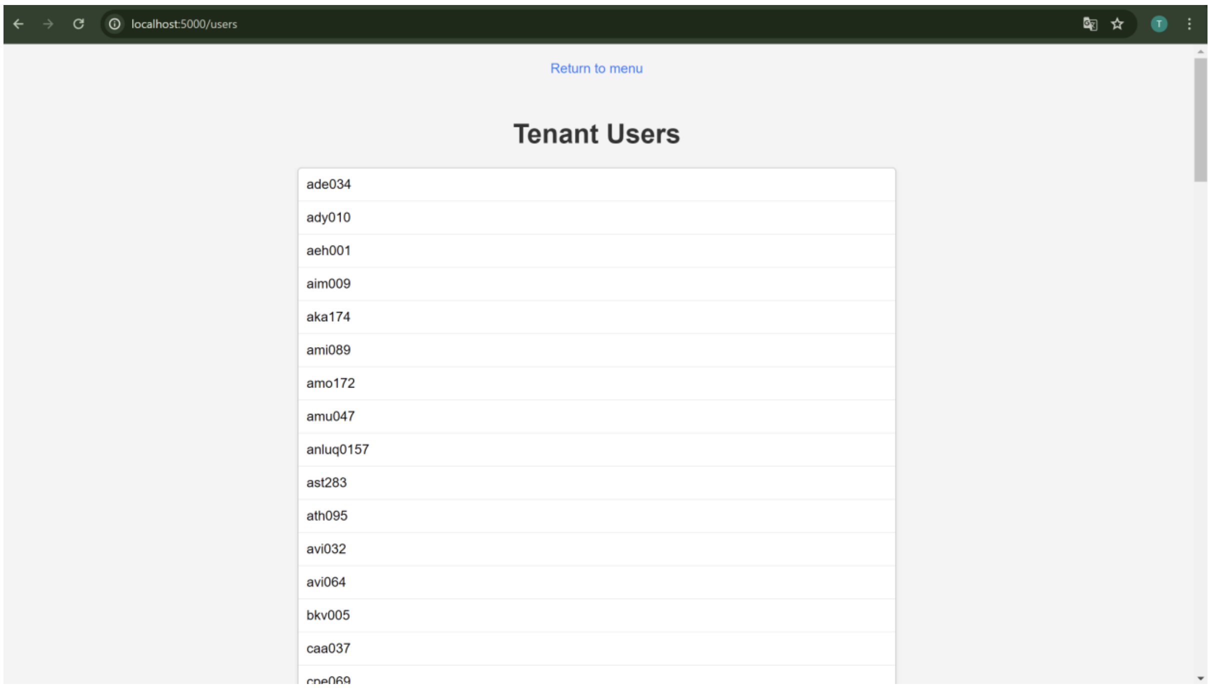
If a user signs in, they are redirected to the personal landing page:


localhost:5000

Welcome tbu025!

Show Users
Edit Profile
Logout

Now, they also have access to the previously unavailable endpoint:


localhost:5000/users

Return to menu

Tenant Users

ade034
ady010
aeh001
aim009
aka174
ami089
amo172
amu047
anluq0157
ast283
ath095
avi032
avi064
bkv005
caa037
cpe069

4.3 Deployment

Even though the web application works as intended, the assignment is not completely successful. One part of the task was to deploy the web application on the provided virtual machines, and not only on localhost.

5 Conclusion

The goal of this project was to implement a web application that authorized users using Microsoft's OpenID. This was achieved. However another part of the task was to deploy the web application, which was not successfully done.

6 Works cited

jwt.io. (2024, April 25). Hentet fra https://jwt.io/introduction

Microsoft. (2024, April 25). *List Users API*. Hentet fra https://learn.microsoft.com/en-us/graph/api/user-list?view=graph-rest-1.0&tabs=http

Microsoft. (2024, April 25). *Update User API*. Hentet fra https://learn.microsoft.com/en-us/graph/api/user-update?view=graph-rest-1.0&tabs=http