Ecosystem infrastructure for smart and personalised inclusion
and PROSPERITY for ALL stakeholders

IAM and integrated applications:

User guide

Gianna Tsakou (gtsakou@singularlogic.eu)

Andreas Papadakis (apapadakis@ep.singularlogic.eu)

Panagiotis Athanasoulis (pathanasoulis@ep.singularlogic.eu)

Table of Contents

List of figures

List of tables

1. IAM component

Identity and Access Management (hereinafter referred to as IAM) refers to the management of user identities and their access permissions to registered resources according to the policies of the resource owners). IAM supports the registration, authentication and authorization of users and resources. The Security component, which offers the IAM functionality for the P4All, is being designed to support all components/services under development in P4All. The core functionality of the IAM component is based on the open source ForgeRock OpenAM tool. Some of the protocols that are used are the OAuth2[1] and the OpenID[2].

Figure 1 depicts an abstract view of the OAuth2 flow. In our case, we typically use term resource to refer any application. Furthermore the role of resource owner can be assigned to the administrator or the developer of the application.

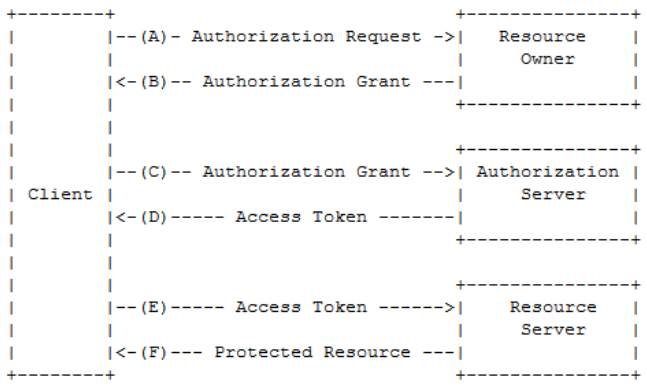

Figure 1 Abstract protocol OAuth2 flow

1.1 Type of users

In the case of the P4All project, we can identify two types of users:

- the *resource owner*, and
- the *end-user* (or *resource consumer*).

The direct user of the Security-IAM component is the *resource owner* (or the developer) of each P4All platform/service (for example AoD, DSpace, Unified List etc.) as he/she allows integration of this platform with the Security-IAM component in order to employ the IAM functionality. The *end-users* of each P4All platform or service (which has been integrated with the IAM component) indirectly use the IAM component according to the rules and peculiarities set by this platform.

[1] https://tools.ietf.org/html/rfc6749
[2] https://openid.net/specs/openid-authentication-2_0.html

1.2 IAM User Interface

The landing page of the IAM component is the link http://83.235.169.221/prosperity4all/identity-access-manager/login/ (Figure 2). There, the user can login on it using the personal credentials or to create a new account.

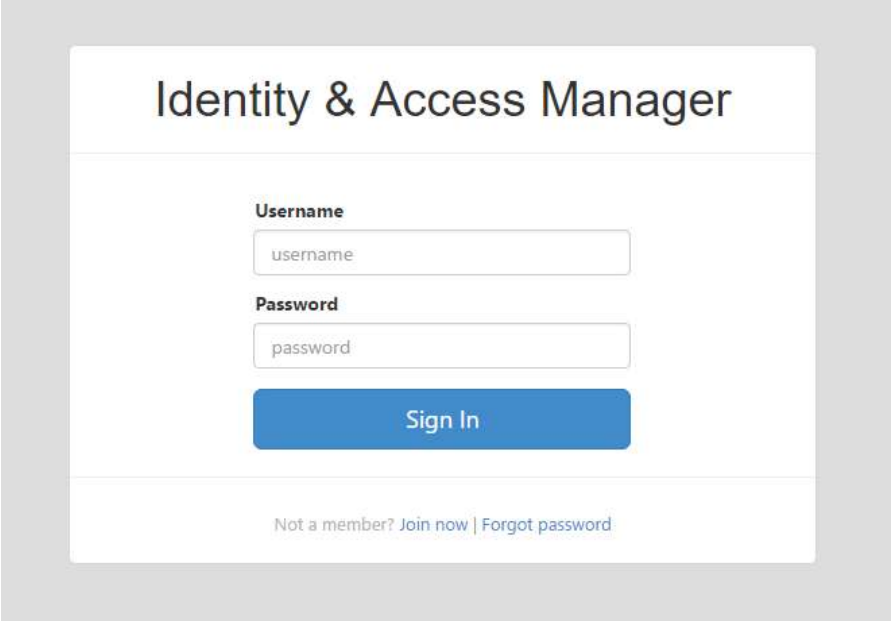

Figure 2 IAM login form

2. Guidelines for resource owners

Any resource owner has to follow the steps below in order to protect any resource server (i.e. a web application) with the IAM component:

- Create an new account
- Access and manage an existing account
- Register the resource server/web application
- Configure the supported roles of web application
- Integrate application with IAM component

2.1 Create account

The resource owner of the application has to create a new account in the IAM component by clicking the "*Join now*" link in the IAM landing page or by following the link http://83.235.169.221/prosperity4all/identity-access-manager/signup-request/. There, the resource owner should be type a valid email account that owns.

When the resource owner submits the email account, an email will be sent in his email account including guidelines how to complete the registration; he/she has to follow the link "*Signup now*" enclosed in the email body to be registered. Figure 3 depicts the fields that the registration form includes. Note that some of the fields are mandatory.

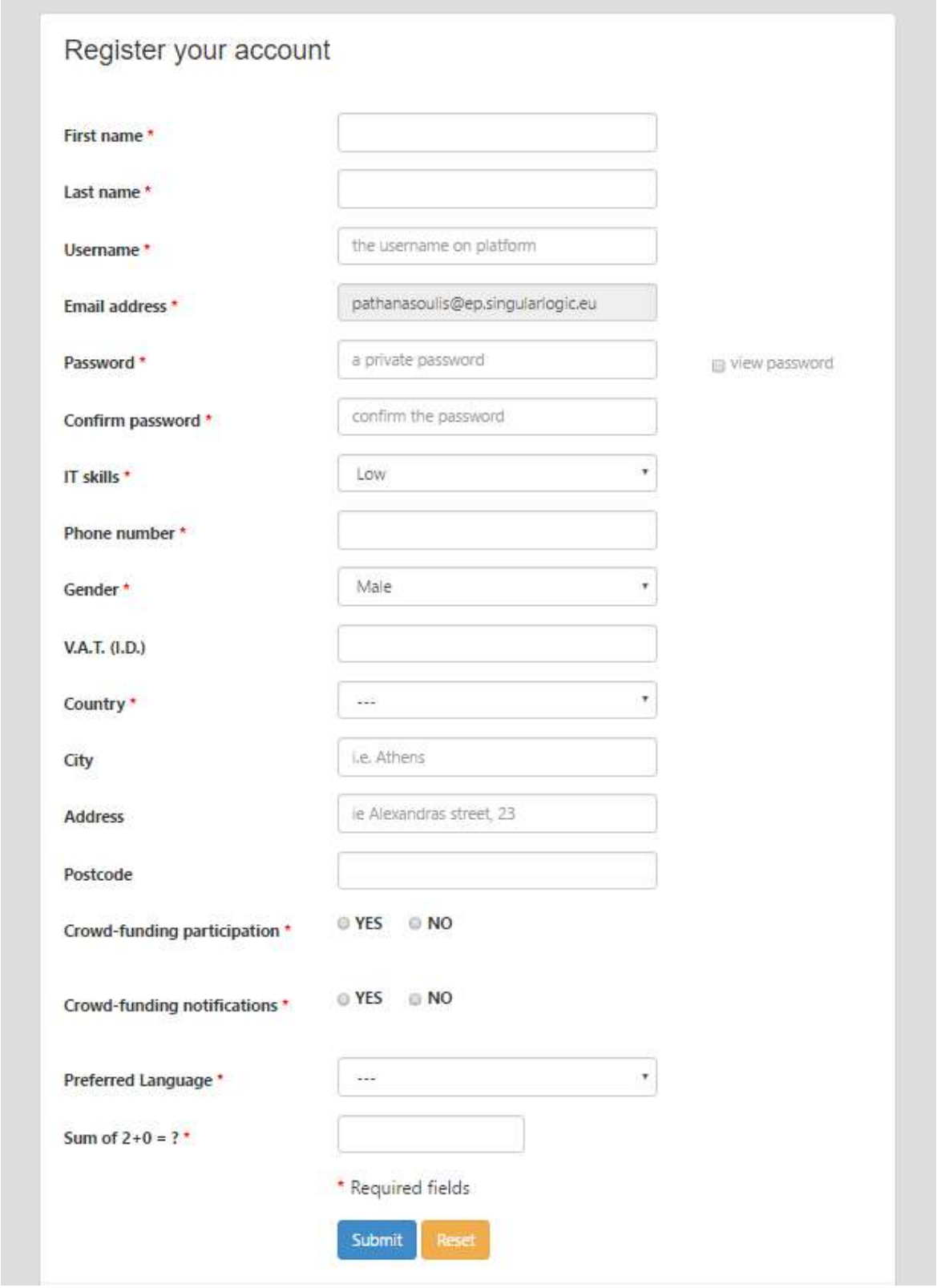

Figure 3 Fill in the required fields in the registration form

2.2 Manage account

2.2.1 Access account

After the successful user authentication, the resource owner can access the overview of his/her profile via the top menu "*Hi {username}* > *My account*". Figure 4 depicts how the profile looks like.

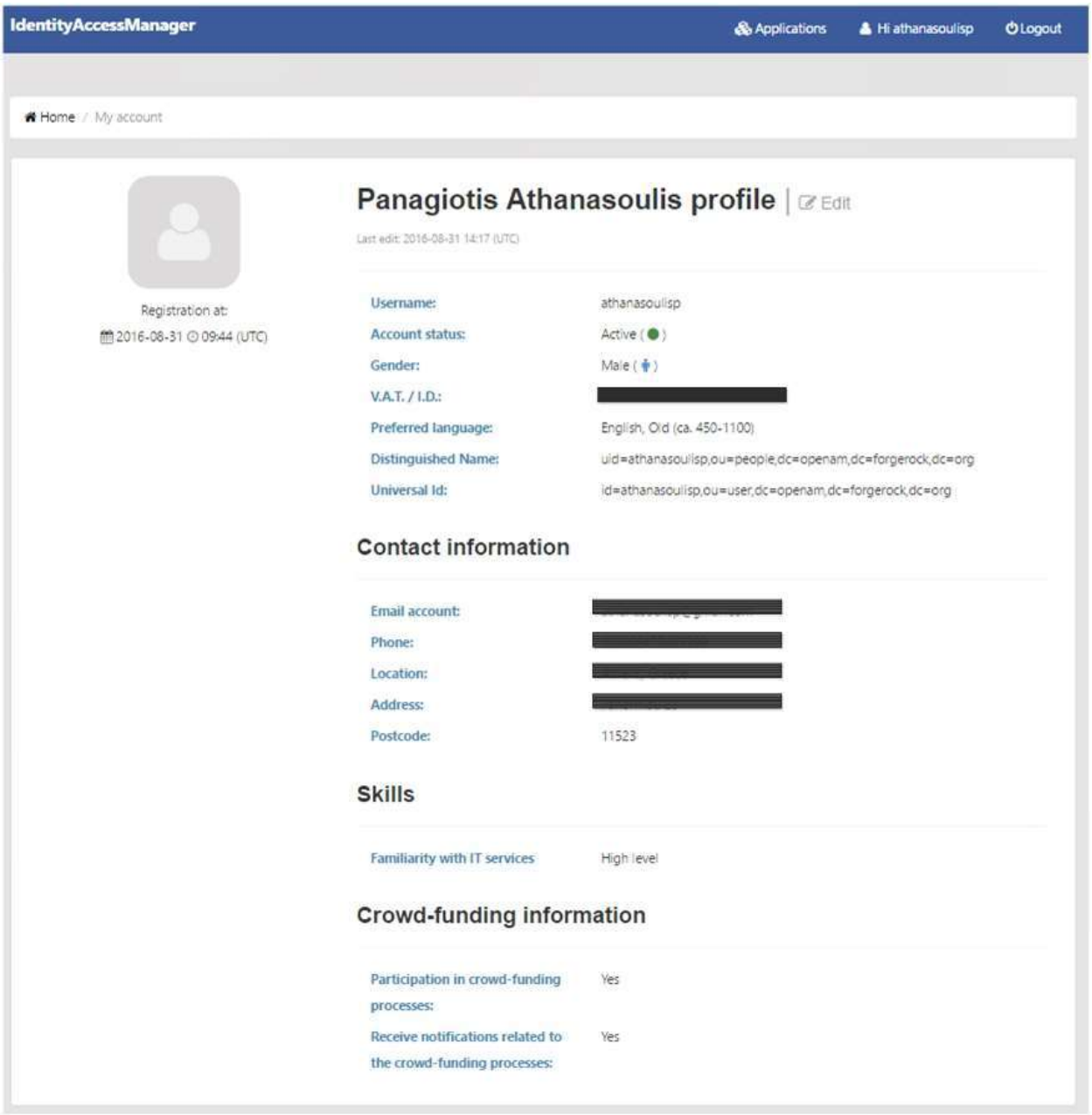

Figure 4 Preview of user profile

2.2.2 Edit account

The resource owner has the capability to update his/her profile (apart from the username) by clicking on "Edit" button as seen in Figure 4.

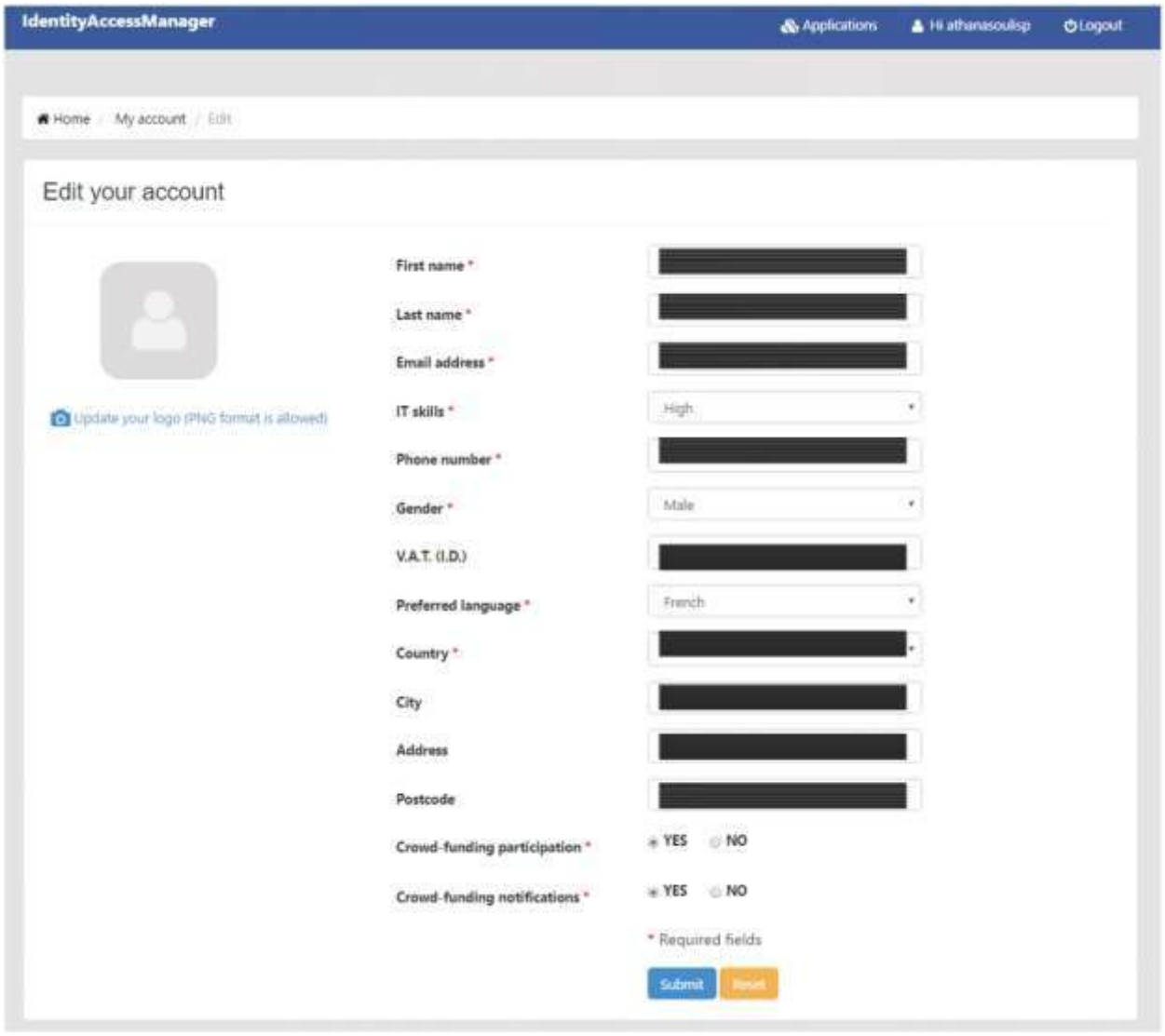

Figure 5 Resource owner update his/her account

2.2.2.1 Change password

Any logged in user has the option to change his/her password as seen in Figure 6. Since resource owner submits the new password, he/she will be directed to the login page to enter the new credentials.

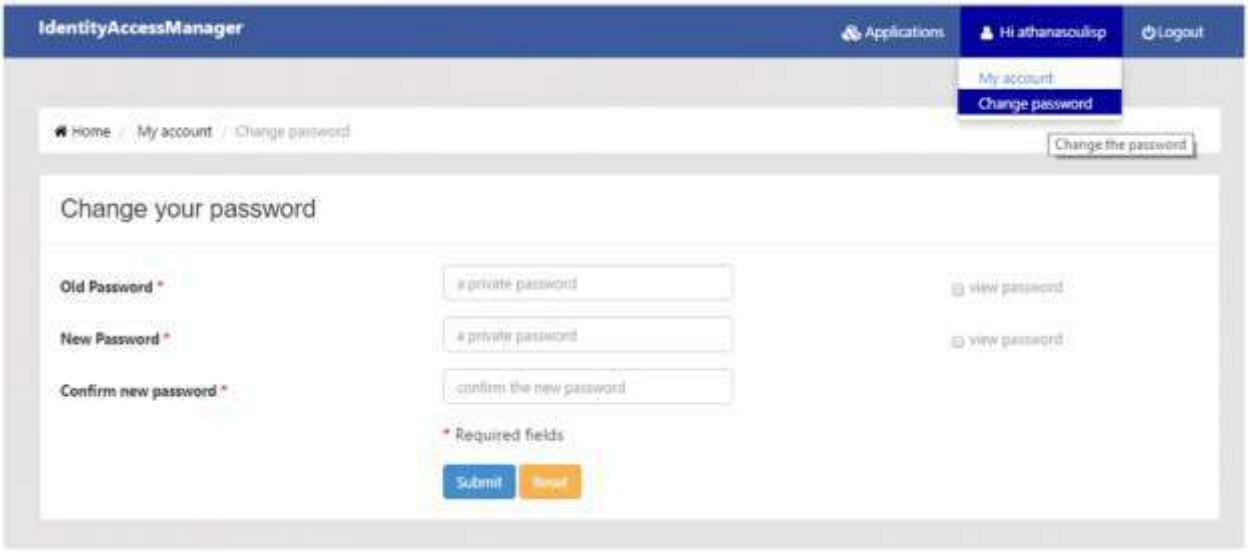

Figure 6 Change the password

2.2.3 Disable account

Service owner must contact with the IAM administrator in order to deactivate the account.

2.2.4 Delete account

Service owner must contact with the IAM administrator in order to delete the account.

2.3 Register a new application

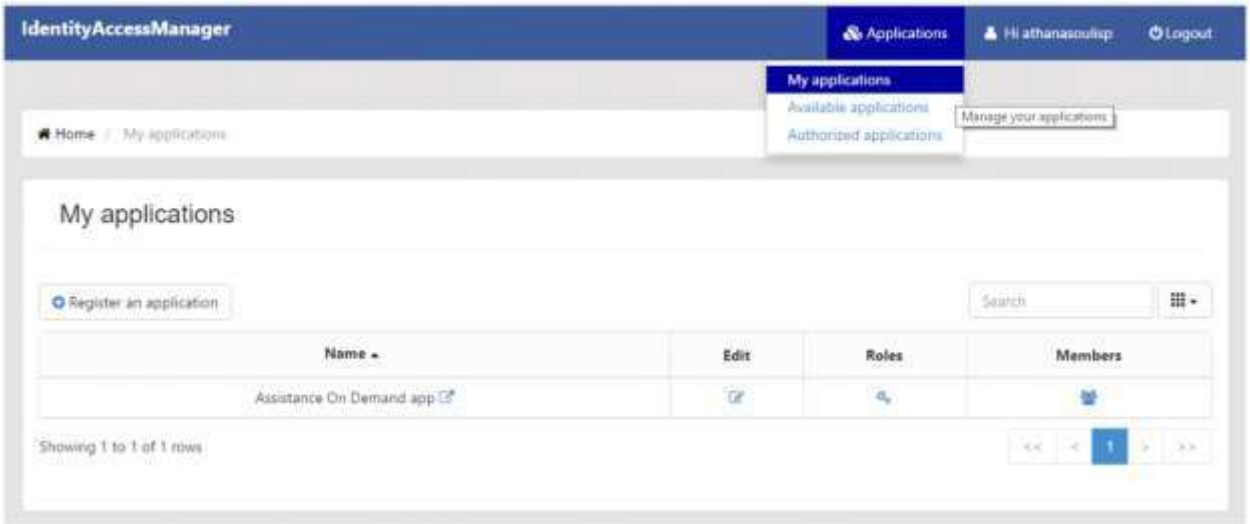

Figure 7 Applications management

Let's suppose that the resource owner has already created an account in IAM and are logged in. Resource owner needs to navigate on the "*Applications > My applications"* (top menu) in order to access his/her applications' dashboard (see Figure 7). This is the place that the resource owner has to register an application as a first step to protect it using the IAM component (for integration purpose).

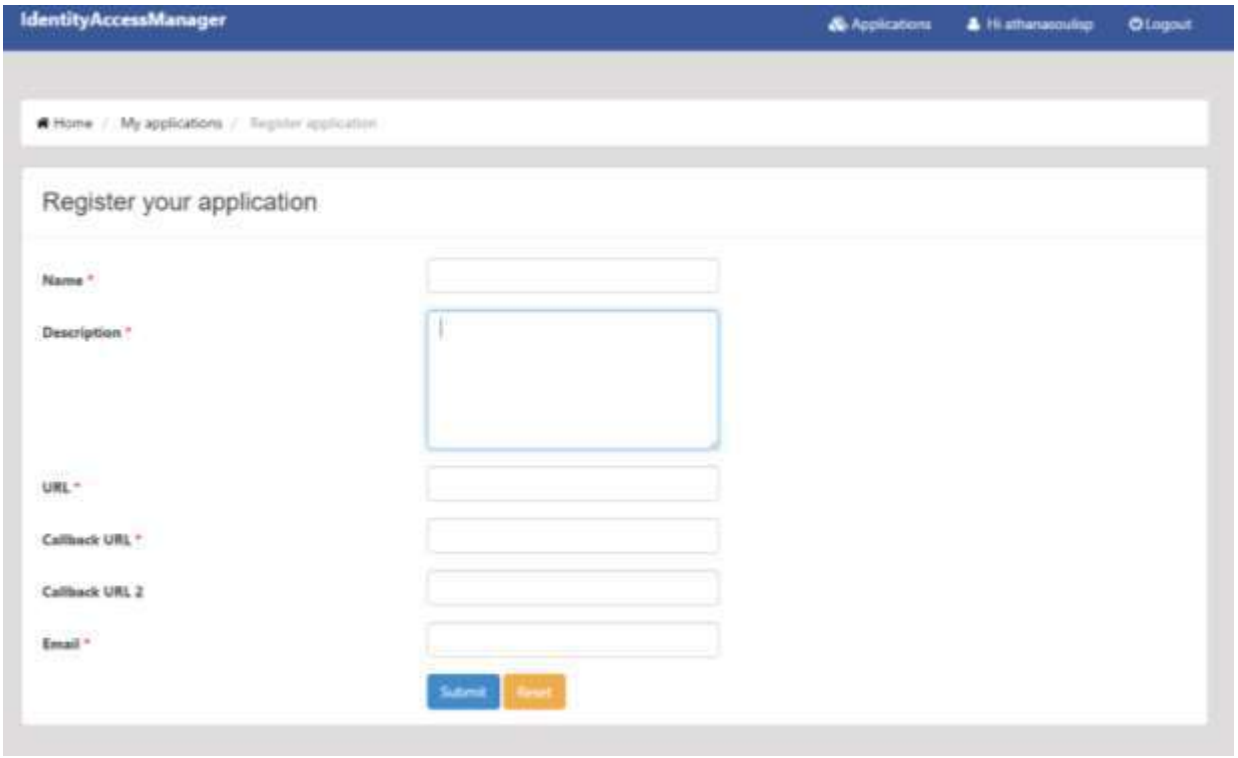

Figure 8 Register a new application

Press the button "*Register an application"* and fill in the required fields of the form (see Figure 8). Enter the name of the application, a brief description of it, the base URL, the callback URL of the application that is required from the oauth2 protocol and a valid email account of the application's administrator (or representative). In meanwhile, the resource owner will be notified about the **CLIENT_ID** and the **CLIENT_SECRET** that characterizes uniquely the registered application in IAM (OpenAM authorization server); it is something like the username and password of the application, so the CLIENT_SECRET must be secret and protected. Figure 9 depicts the details of the *Assistance on Demand application* that has been registered in IAM.

The resource owner can update the details of any registered application that owns by clicking on the "*Edit" button*.

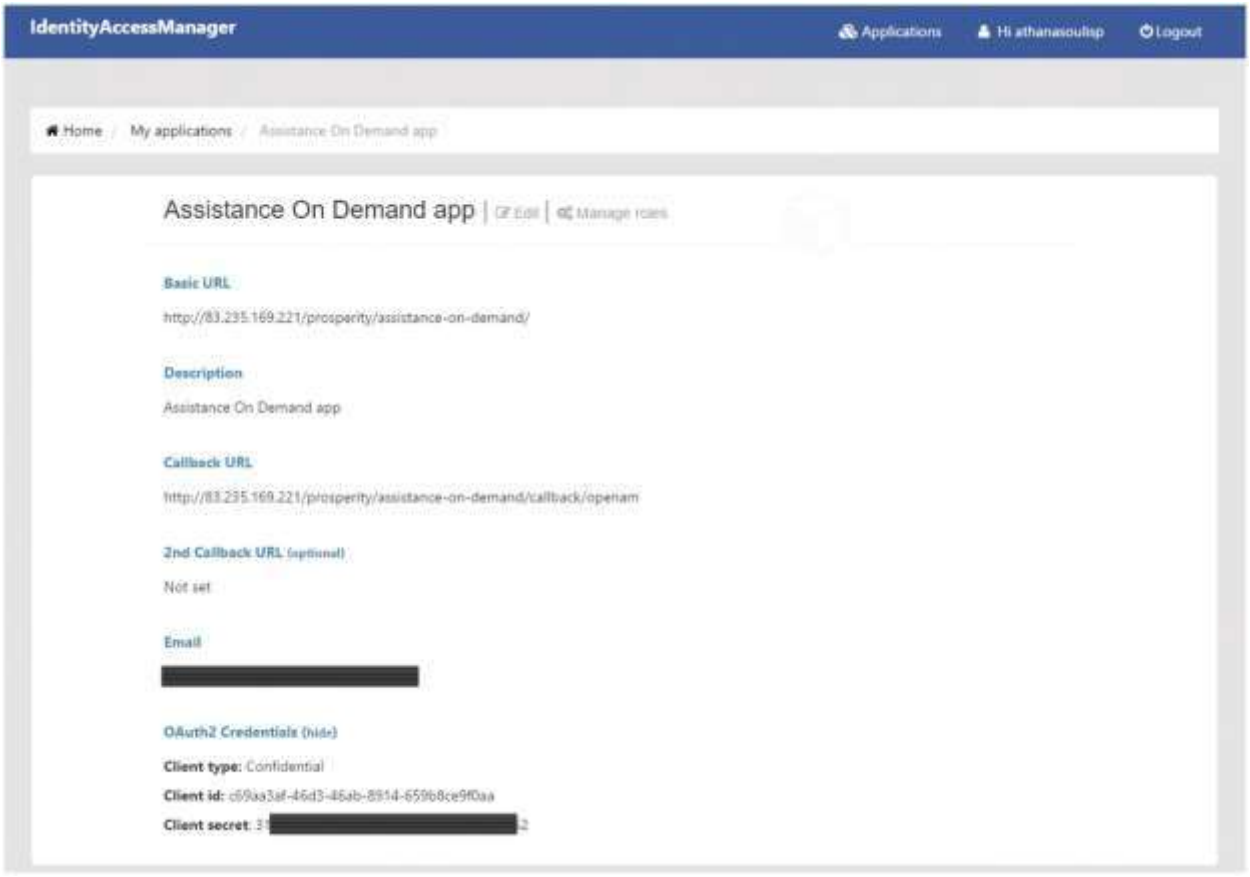

Figure 9 Overview of the Assistance on Demand application's details in IAM

2.4 Set roles in application

The resource owner can manage its roles by clicking on the "*Manage roles*" button (see Figure 9). In case that the desired role does not exist in the list, the resource owner could add it by clicking on "*Add*" *button* and providing the name and a brief description of the role. Alternatively, the resource owner could contact with the IAM administrator.

Figure 10 depicts the roles that the Assistance on Demand application supports (i.e. carer, service_consumer, service_provider).

The IAM component will share on any registered application the roles of any authenticated user but will not perform any policy per role in the application side; each application **is responsible to manage these roles**.

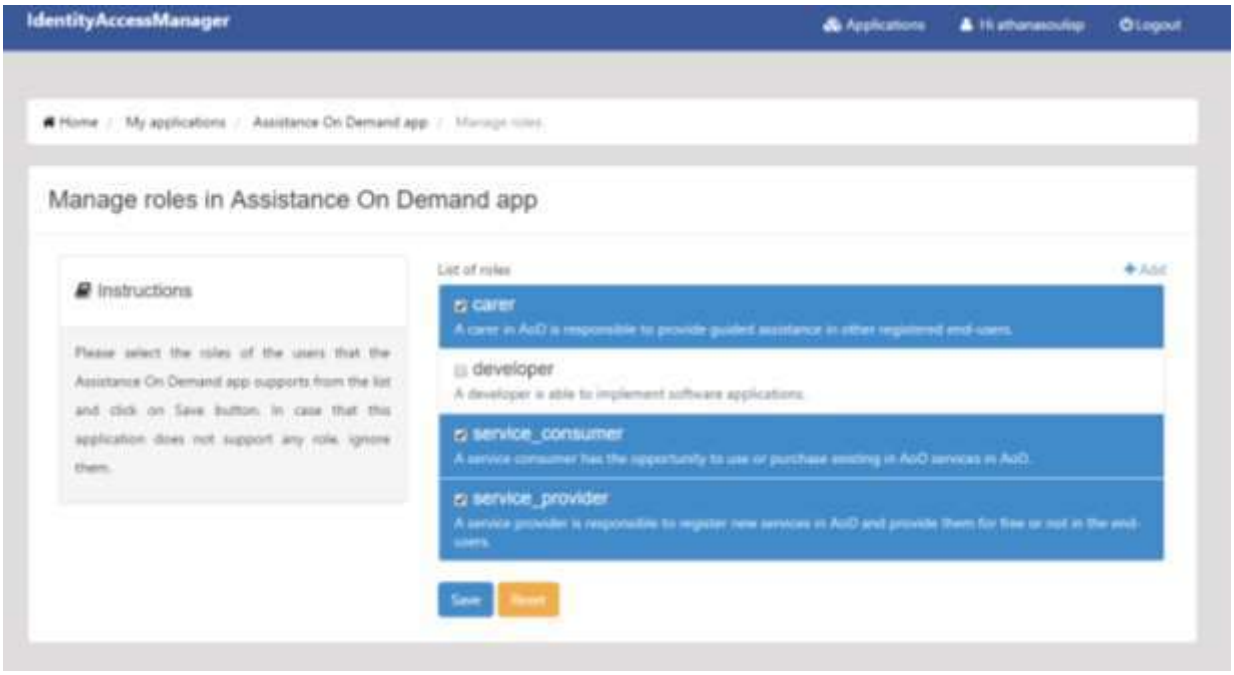

Figure 10 Manage the roles of the Assistance on Demand application

Therefore, a new application has been registered in the IAM component and its roles have been defined from the resource owner.

2.5 Add business logic in the application side

A piece of business logic must be implemented/added in the application side in order to be achieved integration among any (web) registered application and IAM component.

IAM component provide services on application that:

- Authenticate the user
- Authorize the user
  - Retrieve the access token
  - Retrieve the basic or extended user profile
  - Retrieve the roles of user
- Manage the access token
  - Validate the access token
  - Update the access token

The base URL {I**AM_BASE**} of the deployed IAM is: **http://83.235.169.221/.**

2.5.1 User authentication

Firstly, any application protected by IAM component has to redirect any end-user that requires access on it to the IAM login service for authentication. Therefore, the user's browser must be redirected to a URL with the following pattern:

```
GET http://83.235.169.221/prosperity4all/identity-access-manager/oauth2/authorize/?response_type=code&client_id={CLIENT_ID}&redirect_uri={CALLBACK_URL}
```

where the **CLIENT_ID** value has been generated during the registration of the application and characterizes it uniquely, and the **CALLBACK_URL** value has been provided from the application owner during the registration and is provided from the application. These parameters are known both in IAM component and application side. There, the user must enter his/her credentials and submit them. The IAM component receives the request and evaluates the credentials. Since the user has entered valid credentials, IAM generates the **AUTHORIZATION_CODE** and redirects the user's browser to the CALLBACK_URL providing the CODE as a query parameter. The redirect URL must have the following pattern

```
GET {CALLBACK_URL}?code={CODE}
```

2.5.2 User authorization

2.5.2.1 Application requests an access token

The application receives the authorization code since the user has redirected to **CALLBACK_URL**. Therefore, the application can request an access token from IAM component using the **CLIENT_ID**, **CLIENT_SECRET**, and **AUTHORIZATION_CODE** received. The IAM component validates the application credentials and authorization code, and returns an access token to application. To achieve it, the application invokes the web service that is described in Table 1.

The web service returns a JSON object. It includes the access token, the refresh token and the expiration period in seconds. The application should store at least the access token for future usage either in the database or in another type of storage. It is suggested to keep both the access and refresh tokens encrypted.

Table 1 Web service that retrieves an access token

HTTP method	POST
Endpoint	{IAM_BASE}/openam/oauth2/access_token
HTTP headers	Accept: application/json Content-Type: application/x-www-form-urlencoded Authorization: Basic {token} -> encode_base64(CLIENT_ID:CLIENT_SECRET)
Payload	grant_type=authorization_code&code={AUTHORIZATION_CODE}&redirect_uri={CALLBACK_URL}

Normal Response	Status: 200 OK
	
	{
	"scope": "email openid profile",
	"expires_in": integer,
	"token_type": "Bearer",
	"refresh_token": "string",
	"id_token": "string",
	"access_token": " string "
	}

Erroneous response	Status: 400 BAD REQUEST
	
	{
	"error": "invalid_grant",
	"error_description": "The provided access grant is invalid, expired, or revoked."
	}

2.5.2.2 Application retrieves the basic user profile

Since the access token of the user is known and stored, the application could retrieve the basic user profile using the web service in the Table 2. If the access token is valid, the web service returns a JSON object including the username of user (sub field), his/her email, first name and surname.

Table 2 Web service that retrieves the basic user profile

HTTP method	GET
Endpoint	{IAM_BASE}/openam/oauth2/userinfo
HTTP headers	Accept: application/json Content-Type: application/json Authorization: Bearer {access_token}
Payload	-

Normal Response	Status: 200 OK
	
	{
	"sub": "string",
	"updated_at": "string",
	"email": "email",
	"name": "string",
	"family_name": "string",
	"given_name": "string"
	}

Erroneous response	Status: 400 BAD REQUEST { "error": "server_error",

	"error_description": "Access Token not valid" }

2.5.2.3 Application retrieves the extended user profile

Since the access token of the user is known and stored, the application could retrieve the extended user profile using the web service that is described in Table 3. If the access token is valid, the web service returns a JSON object including the username, email, phone, residence details, familiarity with it services (IT skills) and crowd-funding details.

Table 3 Web service that retrieves the extended user profile

HTTP method	GET
Endpoint	{IAM_BASE}/prosperity4all/identity-access-manager/api/oauth2/userinfo?access_token={access_token}
HTTP headers	Accept: application/json
Payload	-

Normal Response	Status: 200 OK
	
	{
	"name": "string",
	"surname": "string",
	"gender": "enum(M, W)",
	"username": "string",
	"country": "string",
	"city": "string",
	"address": "string",
	"postcode": "string",
	"mail": "email",
	"phone": "string",
	"skills": "enum(low, normal, high)",
	"crowd_fund_participation": "boolean",
	"crowd_fund_notification": "boolean"
	}

Unauthorized response	Status: 401 UNAUTHORIZED
	
	{
	"reason": "Access Token is not valid",
	"code": 401
	}

2.5.2.4 Application retrieves the roles of user

Since the access token of the user is known and stored, the application could retrieve the roles of user providing the access token and the CLIENT_ID. To perform it, the web service that is described in Table 4 is used. If the access token is valid, the web service returns a list of JSON objects including the application's CLIENT_ID and name, and the name of role (application_role.role.type).

Table 4 Web service that retrieves the roles of user per application

HTTP method	GET
Endpoint	{IAM_BASE}/prosperity4all/identity-access-manager/api/oauth2/roles?access_token={access_token}&client_id={client_id}
HTTP headers	Accept: application/json
Payload	-

Normal Response	Status: 200 OK
	
	[
	{
	"application_role": {
	"application": {
	"client_id": "string",
	"name": "string"
	},
	"role": {
	"type": "string"
	}
	}
	}
	]

Unauthorized response	Status: 401 UNAUTHORIZED
	
	{
	"reason": "Access Token is not valid",
	"code": 401
	}

Erroneous client response	Status: 404 NOT FOUND
	
	{
	"message": "Client_id is not valid",
	"code": 404,
	"reason": "Not found"
	}

2.5.3 Application manages the access token

2.5.3.1 Application validates the access token

The application must check the status (active or not) of the access token using the web service in Table 5. If the access token is active and valid, then the http status of response is 200. In any different case, the access token either has revoked or has expired.

Table 5 Web service that validates an existing access token

HTTP method	GET
Endpoint	{IAM_BASE}/openam/oauth2/tokeninfo?access_token={access_token}
HTTP headers	Accept: application/json Content-Type: application/json
Payload	-

Normal Response	Status: 200 OK
	
	{
	"scope": [
	"email",
	"openid",
	"profile"
	],
	"grant_type": "authorization_code",
	"email": "string",
	"realm": "/",
	"openid": "string",
	"token_type": "Bearer",
	"expires_in": "integer",
	"access_token": "string",
	"profile": "string"
	}

Erroneous response	Status: 400 BAD REQUEST
	
	{
	"error": "invalid_request",
	"error_description": "Access Token not valid"
	}

2.5.3.2 Application refreshes the expired access token

In case that the access token (that the application has stored) is not valid, there are two options with respect to access token refresh. The most obvious option is to redirect the user's browser to the IAM login page and require from the user to enter his/her credentials again.

An alternative option is to try to refresh the access token using the stored refresh token using the web service described in the Table 6. If the refresh token is still valid, the web service returns a new JSON object that includes a valid access token, an updated refresh token and the expiration period of access token in seconds.

Table 6 Web service that refreshes an invalid access token using the refresh token

HTTP method	POST
Endpoint	{IAM_BASE}/openam/oauth2/access_token
HTTP headers	Accept: application/json Content-Type: application/x-www-form-urlencoded Authorization: Basic {token} -> encode_base64(client_id:client_secret)
Payload	grant_type=refresh_token&refresh_token={refresh_token}

Normal Response	Status: 200 OK
	
	{
	"scope": "email openid profile",
	"expires_in": "integer",
	"token_type": "Bearer",
	"refresh_token": "string",
	"id_token": "string",
	"access_token": "string"
	}

Unauthorized response	Status: 401 UNAUTHORIZED
	
	{
	"error": "invalid_client",
	"error_description": "Client authentication failed"
	}

Erroneous response	Status: 400 BAD REQUEST
	
	{
	"error": "invalid_grant",
	"error_description": "grant is invalid"
	}

2.6 AoD – IAM integration

2.6.1 AoD - IAM sequence diagram

Figure 11 depicts the sequence diagram that is followed to integrate the Assistance on Demand with the IAM component.

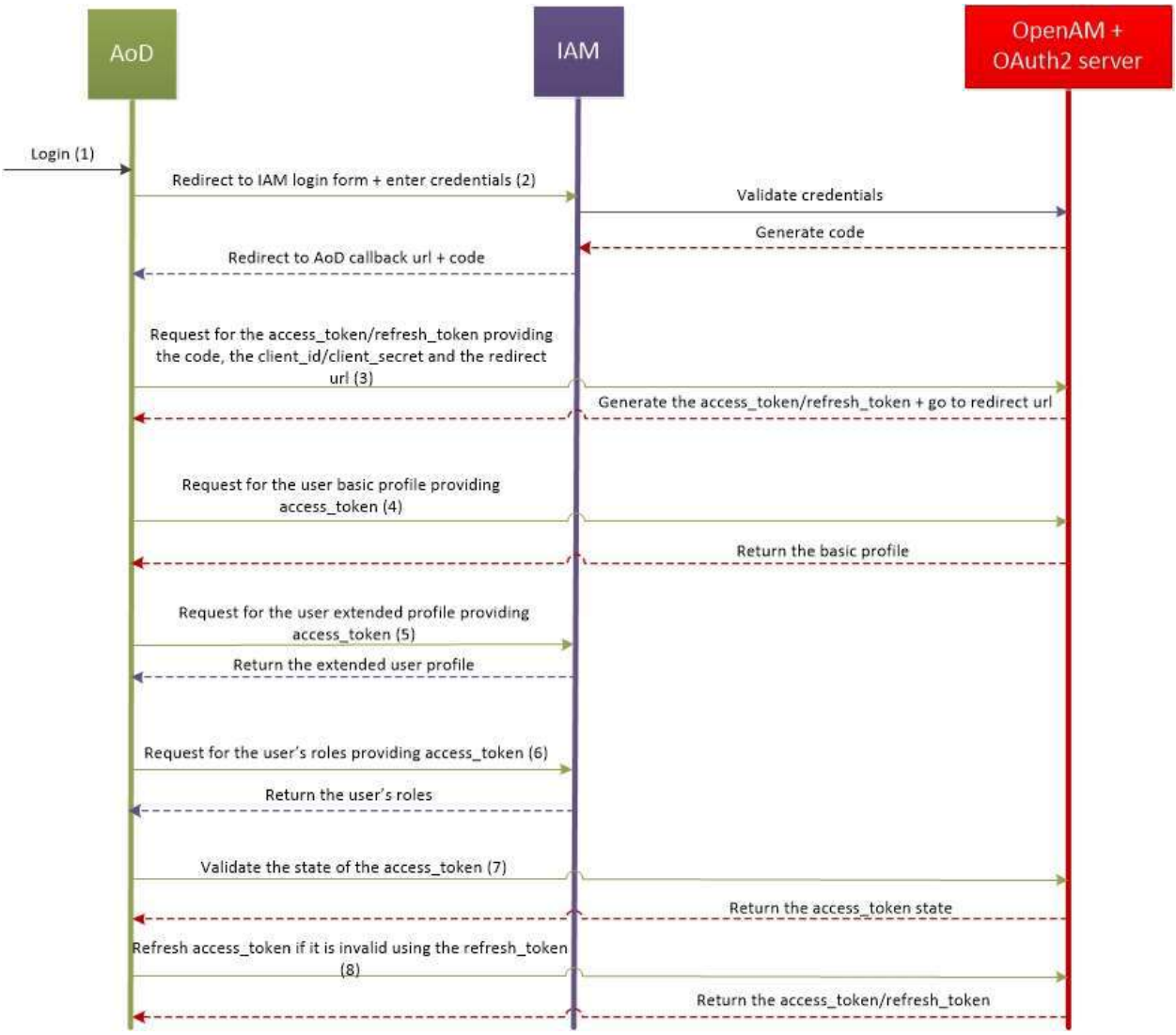

Figure 11 Sequence diagram related to Assistance on Demand app and IAM integration

1. GET http://83.235.169.221/prosperity/assistance-on-demand/login/
2. GET http://83.235.169.221/prosperity4all/identity-access-manager/oauth2/authorize/?response_type=code&client_id={client_id}&redirect_uri={aod_callback_url}
3. POST http://83.235.169.221/openam/oauth2/access_token
4. GET http://83.235.169.221/openam/oauth2/userinfo
5. GET http://83.235.169.221/prosperity4all/identity-access-manager/api/oauth2/userinfo?access_token={access_token}
6. GET http://83.235.169.221/prosperity4all/identity-access-manager/api/oauth2/roles?access_token={access_token}&client_id={client_id}
7. GET http://83.235.169.221/openam/oauth2/tokeninfo?access_token={access_token}
8. POST http://83.235.169.221/openam/oauth2/access_token

2.6.2 How AoD manages the access and refresh tokens

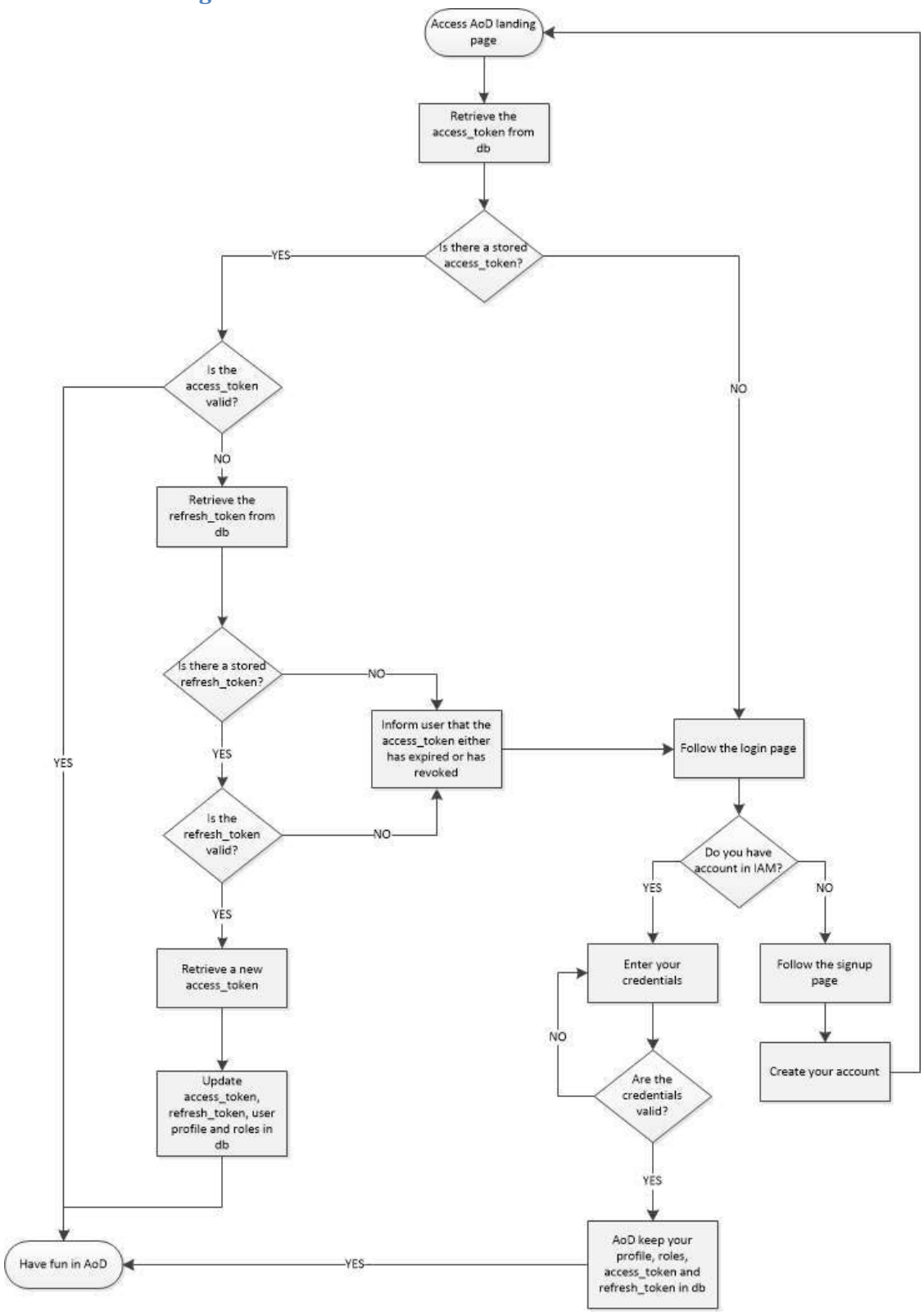

Figure 12 Flow chart that describes how the AoD manage the access tokens

3. Guidelines for end-users

3.1 Create account

Follow the guidelines in section 2.1 Create account.

3.2 Manage account

Follow the guidelines in section 2.2 Manage account.

3.3 Register in an application

Any registered in IAM user can set himself/herself as a member of any registered application in IAM component. An end-user has to navigate via the top menu on *Applications > Available applications* to access the list of registered applications (see Figure 13). The label *owner* in the right side of an application means that the user is also the resource of it. A resource owner can be also member of this application without constraints. Afterwards, the user can navigate on the settings of a specific application by clicking on *Membership* button.

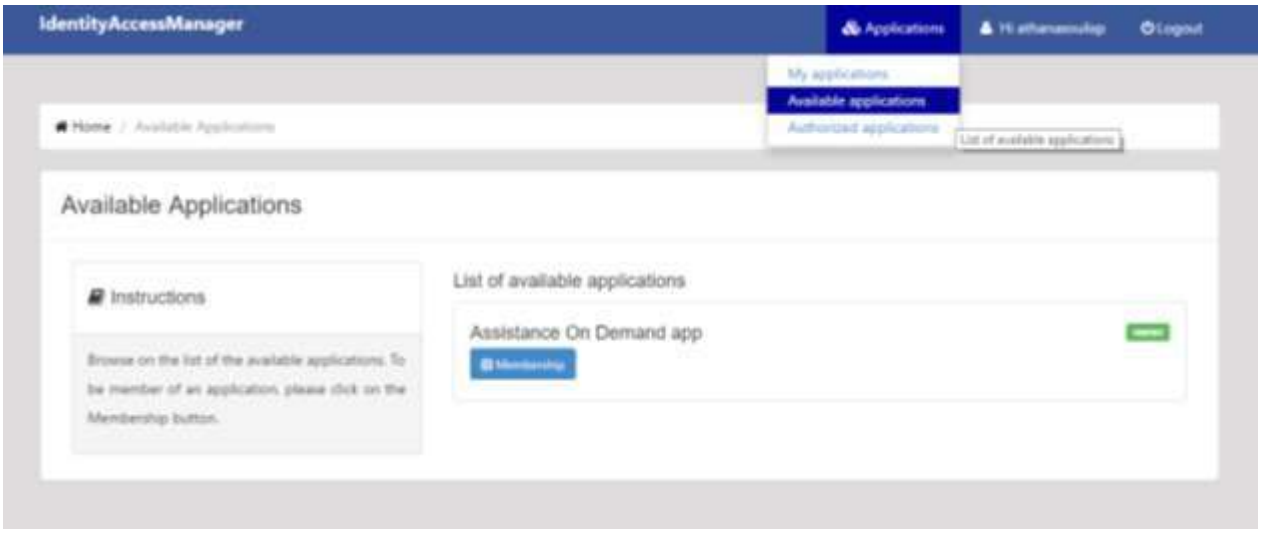

Figure 13 Registered application in IAM component

To become member in an application, the end-user must press YES on the relevant question, read the Terms of usage and submit his/her request (see Figure 14). The IAM component will share a part of user profile (no passwords) in this application. The user has the opportunity to undo his/her choice to be member of a specific application.

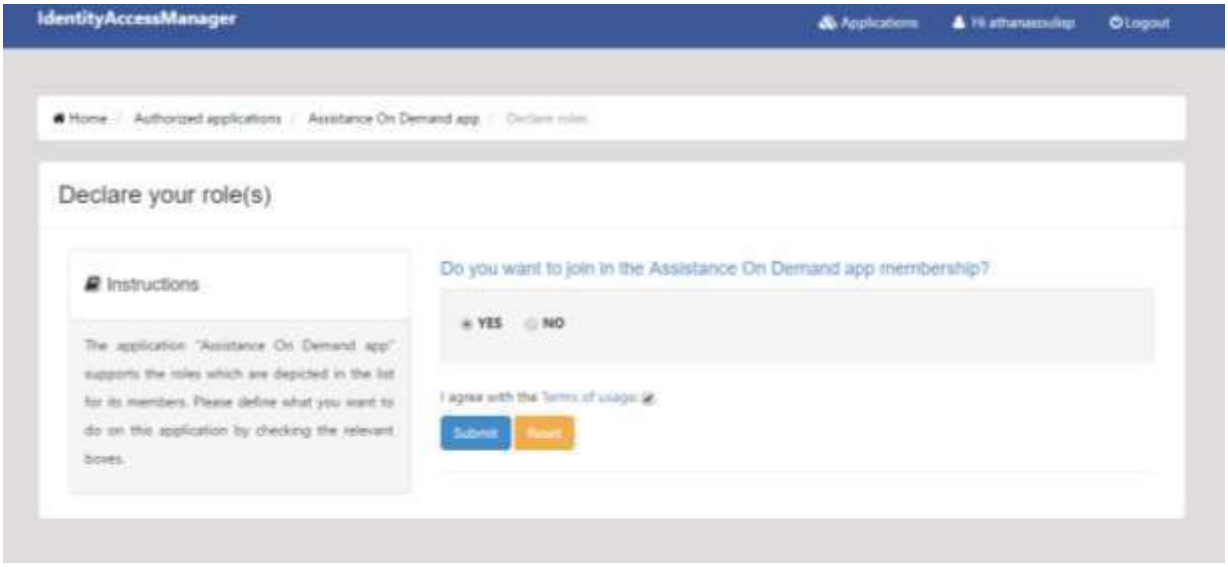

Figure 14 Become member of an application

In the next step, the user must define his/her roles (if exist) in the applications he/she is member. The user accesses a list of supported roles (usually different) per application providing a brief description and is able to select the role(s) of interest. The IAM component will share on this application the roles of the user but each application is responsible to manage the browsing on it.

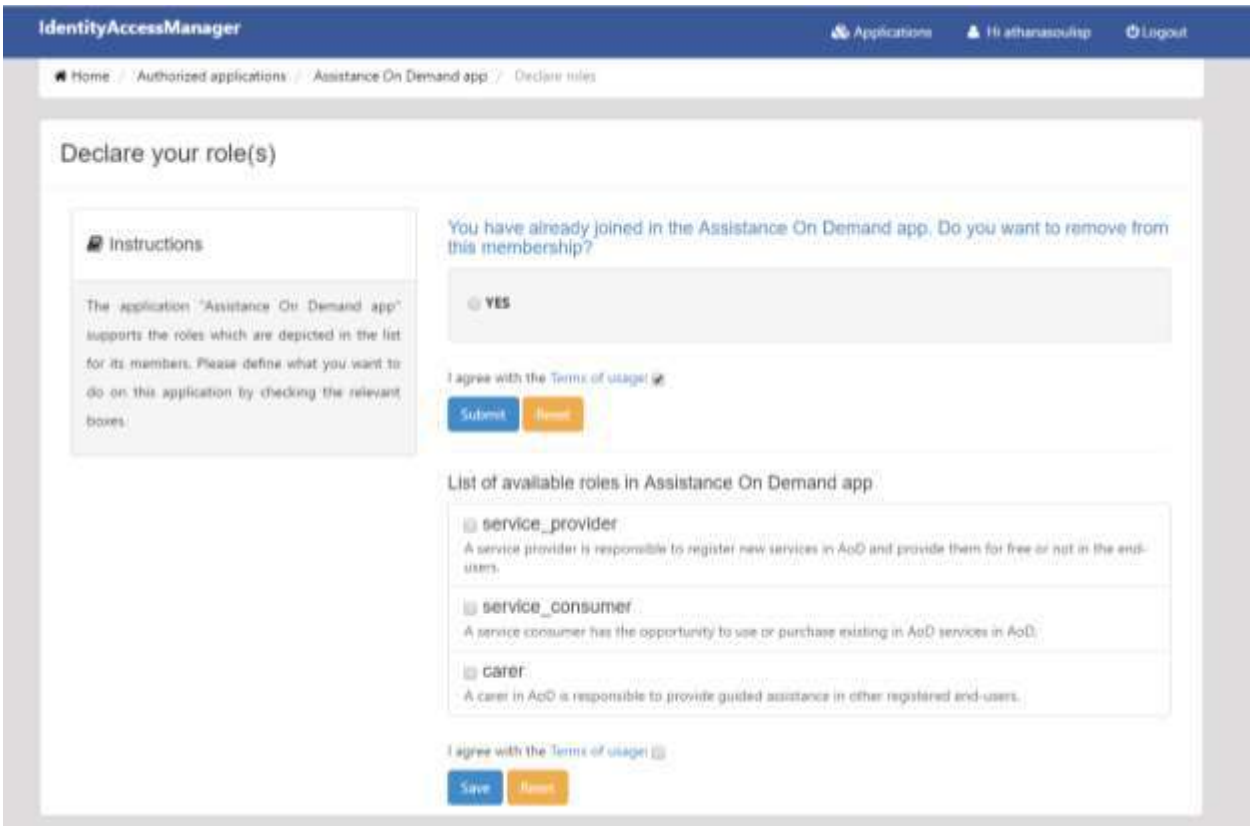

Figure 15 Undo membership in the Assistance on Demand application

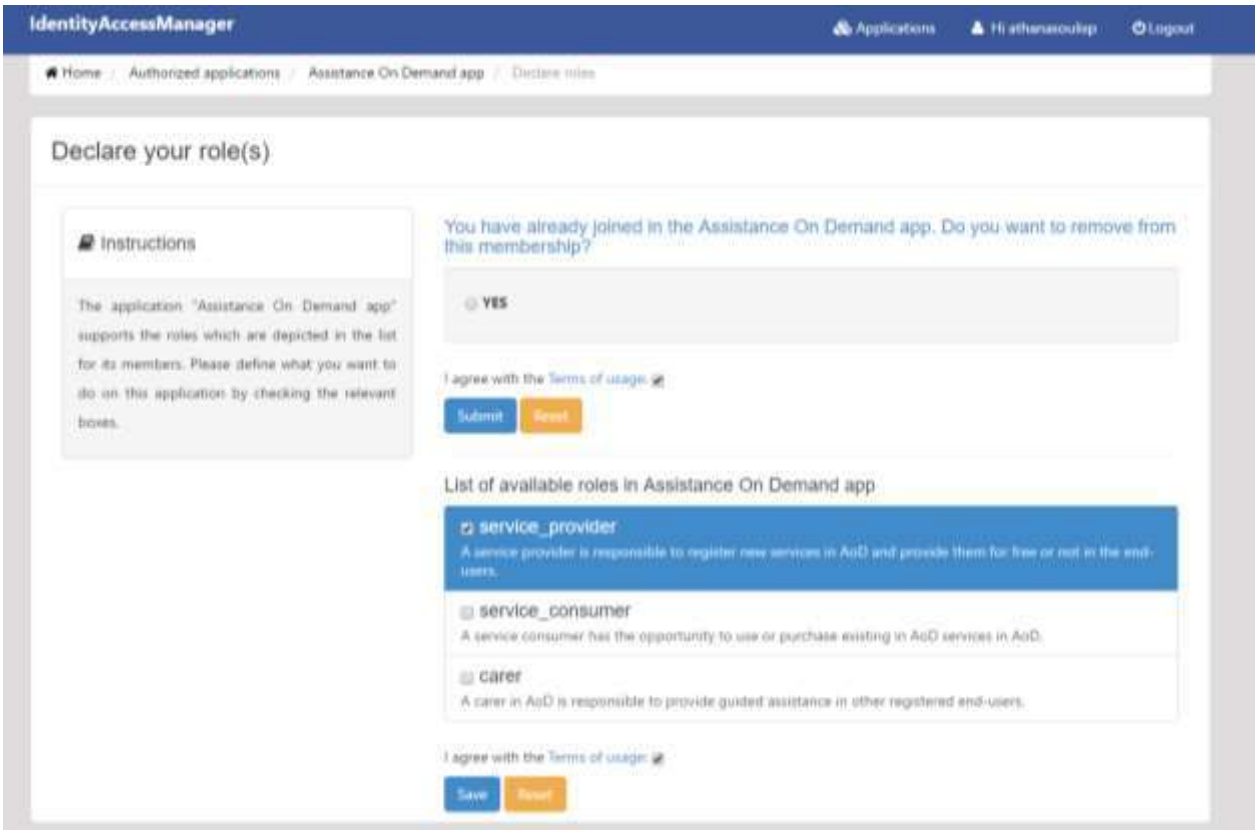

Figure 16 User selects the roles of interest in the Assistance on Demand