

Auth0 setup

Tartalom

Registration	2
Create a custom API	3
Create Application on Applications page Backend	6
Create Application on Applications page Frontend	11
Frontend .env confings	14
Authentication Settings	15
Role handling.....	16
Add administrator user.....	17
Create Actions.....	18
First Action	18
Second action	19
Use Actions.....	20
Login page	22
Create Custom Login Page	24
Translations	25
Social login settings	26
Google	26
Facebook	28
Linkedin	29

SKIPPER - Skills Portfolio of Personal Development (n° 2022-1-HU01-KA220-HED-000086240) project has been funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

Registration

Auth0 registration can be done via various sign-up method. It does not matter what you choose.

Create a custom API

In the left menu click on **Applications --> APIs** and click on **Create API** button. Then fill the inputs as follows:

Create API ✕

Name *

Friendly name for the API. The following characters are not allowed < >

Identifier *

Unique identifier for the API. This value will be used as the `audience` parameter on authorization calls. **Identifier cannot be changed later.**

JSON Web Token (JWT) Profile *

Profile used when issuing `access tokens` for this API.
[Learn more about token profiles](#)

JSON Web Token (JWT) Signing Algorithm *

Algorithm used to sign `access tokens` issued for this API.
[Learn more about signing algorithms](#)

1. Figure Create a new API

Let the **Name** and **Identifier** identical. After you created your custom API you should see two APIs:

 Auth0 Management API System API	API Audience: <code>https://dev-61v6o2j87yp6r1j4.eu.auth0_</code>	<input type="button" value="⋮"/>
 Skipper Custom API	API Audience: <code>Skipper</code>	<input type="button" value="⋮"/>

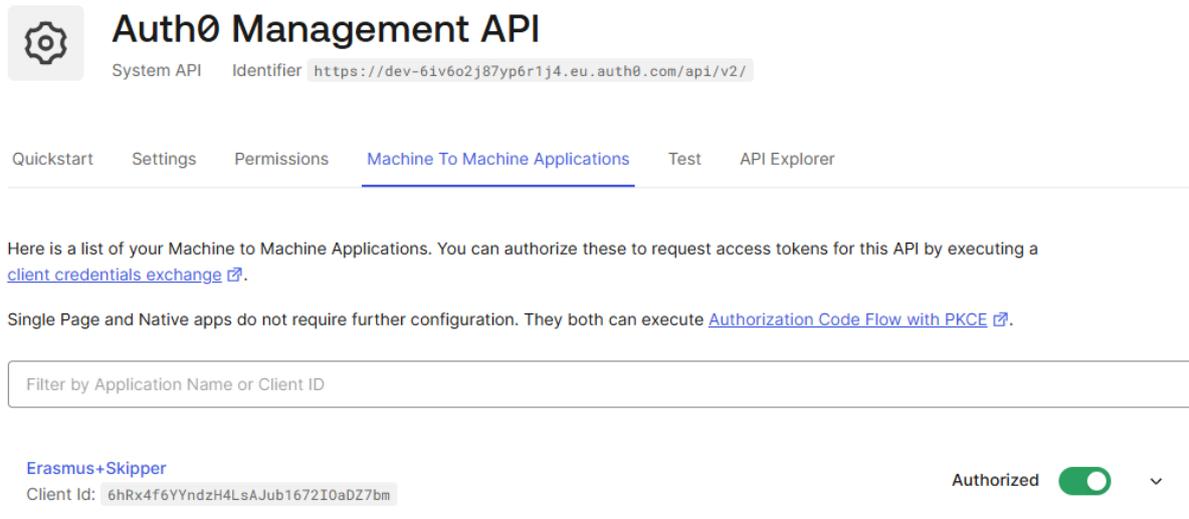
2. Figure Two needed APIs

Skipper API is your own custom API. Auth0 Management API a system generated API which was generated automatically.

Auth0 Management API

Machine To Machine Tab:

Enable the Machine to machine application as authorized.



Auth0 Management API
System API Identifier `https://dev-61v6o2j87yp6r1j4.eu.auth0.com/api/v2/`

Quickstart Settings Permissions **Machine To Machine Applications** Test API Explorer

Here is a list of your Machine to Machine Applications. You can authorize these to request access tokens for this API by executing a [client credentials exchange](#).

Single Page and Native apps do not require further configuration. They both can execute [Authorization Code Flow with PKCE](#).

Filter by Application Name or Client ID

Erasmus+Skipper Client Id: <code>6hRx4f6YyndzH4LsAJub1672I0aDZ7bm</code>	Authorized <input checked="" type="checkbox"/>
--	--

3. Figure Auth0 Management API

Your custom API (Skipper)

Settings tab:

Enable RBAC settings

RBAC Settings

Enable RBAC



If this setting is enabled, RBAC authorization policies will be enforced for this API. [Role](#) and permission assignments will be evaluated during the login transaction.

Add Permissions in the Access Token



If this setting is enabled, the Permissions claim will be added to the access token. Only available if RBAC is enabled for this API.

4. Figure RBAC settings

Access settings:

User Consent Policy ADD-ON

Standard

Policy used when asking users for consent to access their information or perform actions on their behalf. [Learn more about user consent policies](#)

Allow Skipping User Consent



If this setting is enabled, this API will skip user consent for applications flagged as First Party.

Allow Offline Access



If this setting is enabled, Auth0 will allow applications to ask for Refresh Tokens for this API.

5. Figure Access settings

Permissions tab

In **Permissions** tab create Student, Mentor and Admin permissions as follows:

Add a Permission

Define the permissions (scopes) that this API uses.

Permission *	Description *	
<input type="text" value="read:appointments"/>	<input type="text" value="Read your appointments"/>	<input type="button" value="+ Add"/>

List of Permissions

These are all the permissions that this API uses.

Permission	Description	
<input type="text" value="Mentor"/>	Mentor	<input type="button" value="🗑"/>
<input type="text" value="Student"/>	Student	<input type="button" value="🗑"/>
<input type="text" value="Admin"/>	Admin	<input type="button" value="🗑"/>

1. ábra Add permissions

Machine to Machine Applications tab

Enable the newly created application (switch to Authorized). If not yet have then after application is created return to this page to enable.

Erasmus+Skipper

Client Id:

Authorized

2. ábra Enable new API

Create Application on Applications page Backend

To use back-end server with AUTH0 a new application must be created. After you log-in into Auth0 dashboard the following steps are required to make:

1. In the left menu click on **Applications --> Applications** and click on **Create Application** button.

Applications

+ Create Application

Setup a mobile, web or IoT application to use Auth0 for Authentication. [Show more >](#)

3. ábra Create new application

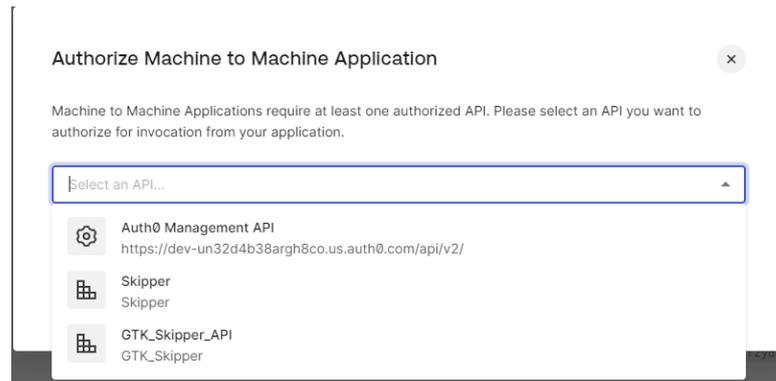
Add a name, select **Machinet to Machine Applications** and click on **Create** button.

Choose an application type

 <p>Native</p> <p>Mobile, desktop, CLI and smart device apps running natively.</p> <p>e.g.: iOS, Electron, Apple TV apps</p>	 <p>Single Page Web Applications</p> <p>A JavaScript front-end app that uses an API.</p> <p>e.g.: Angular, React, Vue</p>	 <p>Regular Web Applications</p> <p>Traditional web app using redirects.</p> <p>e.g.: Node.js, Express, ASP.NET, Java, PHP</p>	 <p>Machine to Machine Applications</p> <p>CLIs, daemons or services running on your backend.</p> <p>e.g.: Shell script</p>
---	--	---	--

4. ábra Machine to Machine Applications

Your custom API must be selected (in example: Skipper) for the new M2M Application. If does not have yet you have to create one before create applications.



5. ábra Select an API for Application

2. Select the newly created API and make the following settings:
 - a. **Settings** tab:

Application URIs section:

- **Allowed Callback URLs:** <http://localhost:3000>,
<http://localhost:3000/dashboard>, <https://aut0-skipper-stage.eu.webtask.run/auth0-authentication-api-debugger>,
<http://localhost:3000/events>

Highlighted part must be your unique tenant domain.

- Allowed Logout URLs: <http://localhost:3000>
- Allowed Web Origins: <http://localhost:3000>

Application URIs

Application Login URI

https://myapp.org/login

In some scenarios, Auth0 will need to redirect to your application's login page. This URI needs to point to a route in your application that should redirect to your tenant's `/authorize` endpoint. [Learn more](#)

Allowed Callback URLs

http://localhost:3000, http://localhost:3000/dashboard, https://aut0-skipper-stage.eu.webtask.run/auth0-authentication-api-debugger, http://localhost:3000/events

After the user authenticates we will only call back to any of these URLs. You can specify multiple valid URLs by comma-separating them (typically to handle different environments like QA or testing). Make sure to specify the protocol (`https://`) otherwise the callback may fail in some cases. With the exception of custom URI schemes for native clients, all callbacks should use protocol `https://`. You can use [Organization URL](#) parameters in these URLs.

Allowed Logout URLs

http://localhost:3000

Comma-separated list of allowed logout URLs for redirecting users post-logout. You can use wildcards at the subdomain level (`*.google.com`). Query strings and hash information are not taken into account when validating these URLs. [Learn more about logout](#)

Allowed Web Origins

http://localhost:3000

Comma-separated list of allowed origins for use with [Cross-Origin Authentication](#), [Device Flow](#), and [web message response mode](#), in the form of `<scheme> "://" <host> [":" <port>]`, such as `https://login.mydomain.com` or `http://localhost:3000`. You can use wildcards at the subdomain level (e.g.: `https://*.contoso.com`). Query strings and hash information are not taken into account when validating these URLs.

Cross-Origin Authentication section:

Enable Cross-Origin Authentication.

Allow Cross-Origin Authentication



When allowed, [cross-origin authentication](#) allows applications to make authentication requests when the Lock widget or custom HTML is used.

Refresh Token Expiration section:

Enable Inactivity Expiration

Set Idle Refresh Token Lifetime



Require refresh tokens to expire after a set period of inactivity.

[Learn more about refresh token expiration](#)

Refresh Token Rotation section:

Enable Rotation.

Allow Refresh Token Rotation



When allowed, refresh tokens will automatically be invalidated after use and exchanged for new tokens to prevent replay attacks. Requires a maximum refresh token lifetime.

[Learn more about refresh token rotation](#)

Advanced Settings section:

On Grant Types tab enable: Implicit, Authorization Code, Refresh Token and Client Credentials

Advanced Settings

Application Metadata

Device Settings

OAuth

Grant Types

WS-Federation

Certificates

Endpoints

Grants

<input checked="" type="checkbox"/> Implicit	<input checked="" type="checkbox"/> Authorization Code	<input checked="" type="checkbox"/> Refresh Token	<input checked="" type="checkbox"/> Client Credentials	<input type="checkbox"/> Password
<input type="checkbox"/> MFA				

Credentials tab

Select **Client Secret (Post)** option and save it.

Authentication Methods
Configure the method to use when making requests to any endpoint that requires this client to authenticate.

Method

Private Key JWT ENTERPRISE Client Secret (Post)

Client Secret (Basic) None

Client Secret

..... @ 📄

The Client Secret is not base64 encoded.

Save Cancel

6. ábra Set authentication method

APIs tab

Enable **Auth0 Management API** and the **created API** (in example: Skipper).

<p>Auth0 Management API</p> <p>API Identifier: <code>https://dev-un32d4b38argh8co.us.auth0.com/api/v2/</code></p>	Authorized <input checked="" type="checkbox"/>	▼
<p>Skipper</p> <p>API Identifier: <code>Skipper</code></p>	Authorized <input checked="" type="checkbox"/>	▼

7. ábra Enable APIs

- Backend env configs

HTTP_PORT=3000

CLIENT_ORIGIN_URL=FRONT_END_ORIGIN_URL

AUTH0_AUDIENCE=Created API name (in example: Skipper)

AUTH0_DOMAIN= Created Application (tenant) Domain from setting tab

MANAGEMENT_CLIENT_DOMAIN= Created Application (tenant) Domain from setting tab

CLIENT_SECRET= Created Application Client Secret from setting tab

CLIENT_ID= Created Application Client ID from setting tab

PDF_WORKER=../pdf/pdfWorker.ts

QUESTIONNARIE_FILE=../../resources/data-service/csv/questionnaires-proofread.csv

MINIO_ENDPOINT=MINIO endpoint

MINIO_PORT=MINIO port

MINIO_USER=MINIO username

MINIO_PASSWORD=MINIO password

MINIO_BUCKET=BUCKET name

MINIO_SSL=true or false

NODEMAILER_HOST=HOST
NODEMAILER_PORT=POST
NODEMAILER_USER=Username
NODEMAILER_PASSWORD=password
NODEMAILER_SSL=true or false

Minio

To store passports, university logo and profile pictures you need to configure a Minio server. If the minio server is configured appropriately you only need to specify the minio env variables and it should work.

Email

To make system able to send e-mail notifications you need to configure the Nodemailer env variables and set up an e-mail service.

The e-mail template can be found under `\src\services\email\templates\` folder. You are free to modify as you wish.

Passport translations

Passports are always translated into application language. The translation files can be found under `resources\locales\` folder. The translation files should place in subfolder named with language code (e.g.: en) and must be named as **translations.json**.

Badges, logos

The badges nad logos are placed under `assets` folder. These images are used during passport generation.

Create Application on Applications page Frontend

- Applications -> Applications menu
- Click on **Create Application** button
- Create new **Single Page Web Application** with react technology

Create application



Name *

My App

You can change the application name later in the application settings.

Choose an application type

 <p>Native</p> <p>Mobile, desktop, CLI and smart device apps running natively.</p> <p>e.g.: iOS, Electron, Apple TV apps</p>	 <p>Single Page Web Applications</p> <p>A JavaScript front-end app that uses an API.</p> <p>e.g.: Angular, React, Vue</p>	 <p>Regular Web Applications</p> <p>Traditional web app using redirects.</p> <p>e.g.: Node.js Express, ASP.NET, Java, PHP</p>	 <p>Machine to Machine Applications</p> <p>CLIs, daemons or services running on your backend.</p> <p>e.g.: Shell script</p>
--	---	---	---

8. ábra Frontend Application létrehozása

Settings tab:

- **Allowed URLs:** <http://localhost:3006>, <http://localhost:3006/dashboard>, <https://dev-un32d4b38argh8co.us.webtask.run/auth0-authentication-api-debugger>, <http://localhost:3006/events>, <http://skipper.example:3006/institution>, <http://localhost:3006/profile-setup>, **your-skipper-domain/dashboard**
- **Allowed Logout URLs:** **<your skipper domain>**, <http://localhost:3006>
- **Allowed Web Origins:** <http://localhost:3006>
- Refresh Token Rotation: On

Frontend .env confings

Create .env file in the frontend application root folder with the following content:

- VITE_AUTH0_DOMAIN: Created Application Domain from setting tab
- VITE_AUTH0_CLIENT_ID: Created Application Client ID from setting tab
- VITE_AUDIANCE: Created API name
- VITE_AUTH0_CALLBACK_URL: <url>/dashboard
- VITE_API_SERVER_URL: <url>/api
- VITE_LOGOUT_URL: <url>
- PORT: 3006 or any other port except the port that the backend uses

Authentication Settings

- **Email-Password** authentication is on by default
- Authentication -> Social
- Click on **Create connection** button
- Add the following options:
 - Facebook
 - Goggle oauth2
 - LinkedIn

Social Connections

+ Create Connection

Configure social connections like Facebook, Twitter, Github and others so that you can let your users login with them. [Show more >](#)

 facebook  Facebook	• 5 Applications enabled	
 google-oauth2  Google / Gmail	• 6 Applications enabled	
 linkedin  LinkedIn	• 5 Applications enabled	

9. ábra Social login options

In every social login disable the possibility of Sync user profile:

Advanced

Sync user profile attributes at each login



Role handling

- User Management -> Roles
- Create the following roles by clicking on **Create role** button:
 - Name: Admin, Description: <Some role description>
 - Name: Mentor, Description: <Some role description>
 - Name: Student, Description: <Some role description>

Roles

+ Create Role

Create and manage Roles for your applications. Roles contain collections of Permissions and can be assigned to Users.

Name	Description	
Admin	Role for admin.	...
Mentor	Role for mentor.	...
Student	Role for student.	...

10. ábra Required roles

For each role please set the appropriate permission on Permission tab of selected role with Add Permission button. The example show this for Admin role:

Admin

Role ID `ro1_2SVdjQtcYsnPDzN7`

Settings **Permissions** Users

Add Permissions to this Role. Users who have this Role will receive all Permissions below that match the API of their login request.

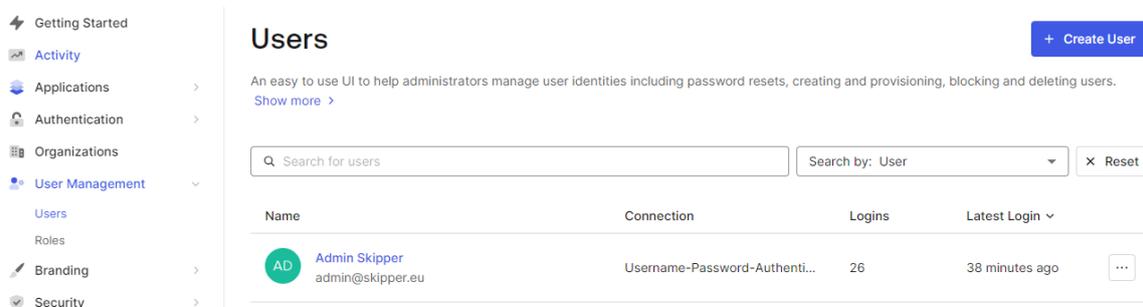
Add Permissions

Permission ^	Description	API	
Admin	Admin	Skipper	🗑️

6. Figure For example: Admin role

Add administrator user

- User Management -> Users
- Click on **Create User** button and add the user information
- After the user creation the new user appear on Users list like on the following example



11. ábra Example user

- Click on ... button in the users table and select **Assign Roles** option
- For admin on **Details** tab and in **User Metadata** section the registered university id (edu_id) must be configured as follows:

Metadata

User Metadata (user_metadata)

```
1 {  
2   "edu_id": "TU01"  
3 }
```

12. ábra Link admin to university

View Details

- Assign Roles
- Assign Permissions
- Send Verification Email
- Change Email
- Change Password
- Block
- Delete

13. ábra ... menu on user row

- In a modal you can add Admin role to the user by selecting the created Admin role

Create Actions

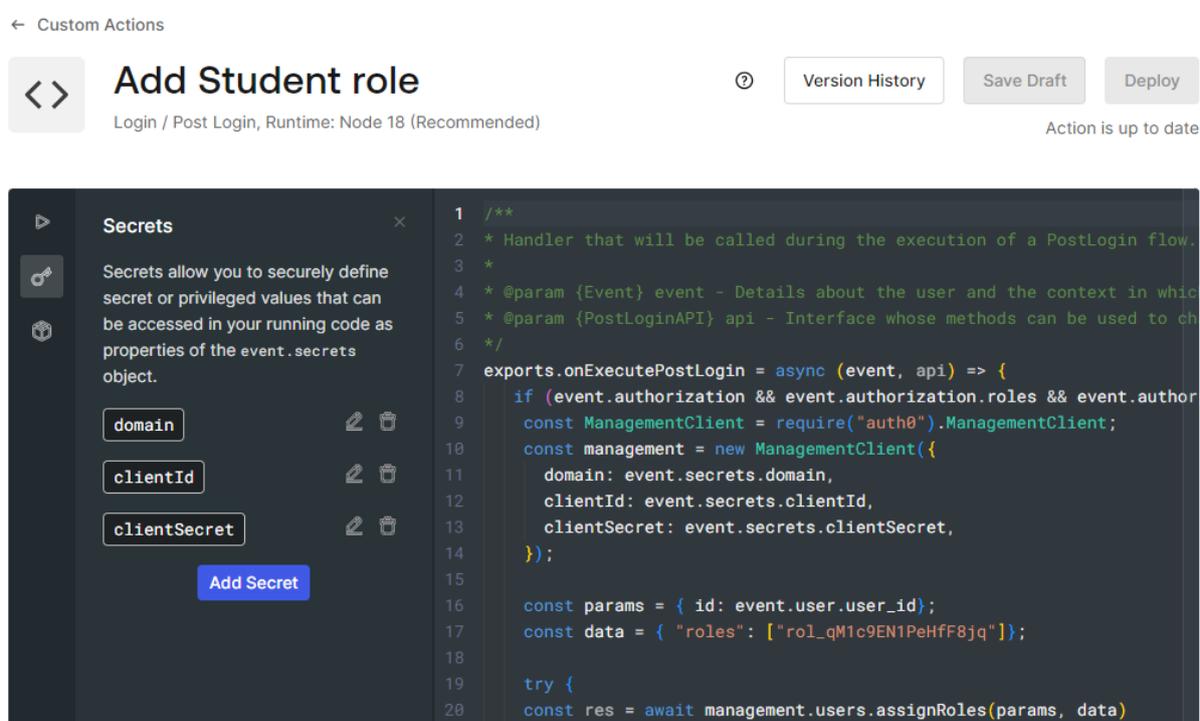
You need to create two custom actions to add default roles to the students automatically and to return user roles in the token on login action.

- Actions -> Library menu
- Click on **Create Action** button and select **Build from scratch** option:

First Action

- Name: Name of your action like: “Add Student role”
- Trigger: Login / Post Login
- Runtime: Node 18

When you have an action, you can define what it should do in a code editor like in the following picture.



14. ábra Add Student role action

Add Secrets

In the left hand side you can add secrets by clicking on the second button with a key in it called “Secrets”.

Add the following keys by clicking **Add Secret** button:

- Key: domain, Value: <your Application domain from setting tab>
- Key: clientId, Value: <your Application Client ID from setting tab>
- Key: clientSecret, Value: <your Application Client Secret from setting tab>

Add dependencies by clicking **Dependencies** button under the Secrets button:

- **Click on Add Dependency**
 - a. Name: auth0
 - b. Version: 4.2.0

Now you can add the following code:

```
exports.onExecutePostLogin = async (event, api) => {  
  if (event.authorization && event.authorization.roles &&  
event.authorization.roles.length === 0) {  
    const ManagementClient = require("auth0").ManagementClient;  
    const management = new ManagementClient({  
      domain: event.secrets.domain,  
      clientId: event.secrets.clientId,  
      clientSecret: event.secrets.clientSecret,  
    });  
  
    const params = { id: event.user.user_id};  
    const data = { "roles": ["<ID of the created Student role>"]};  
  
    try {  
      const res = await management.users.assignRoles(params, data)  
    } catch(e) {  
      console.log(e)  
    }  
  }  
};
```

Now you can click on **Deploy** button to deploy your action.

Second action

Create new action with the name of “Add roles to token” and add the following settings:

Secrets:

- domain

- clientId
- clientSecret

Dependencies:

- Auth0@4.2.0

Code:

```
exports.onExecutePostLogin = async (event, api) => {
  const ManagementClient = require('auth0').ManagementClient;
  const management = new ManagementClient({
    domain: event.secrets.domain,
    clientId: event.secrets.clientId,
    clientSecret: event.secrets.clientSecret,
    scope: 'read:users update:users read:roles'
  });

  const namespace = 'roles';
  if (event.authorization) {
    api.idToken.setCustomClaim(`${namespace}/roles`,
event.authorization.roles);

    api.accessToken.setCustomClaim(`${namespace}/roles`,
event.authorization.roles);

    api.idToken.setCustomClaim(`${namespace}/registry_code`,
event.user.user_metadata.registry_code);

    api.accessToken.setCustomClaim(`${namespace}/registry_code`,
event.user.user_metadata.registry_code);
  }
};
```

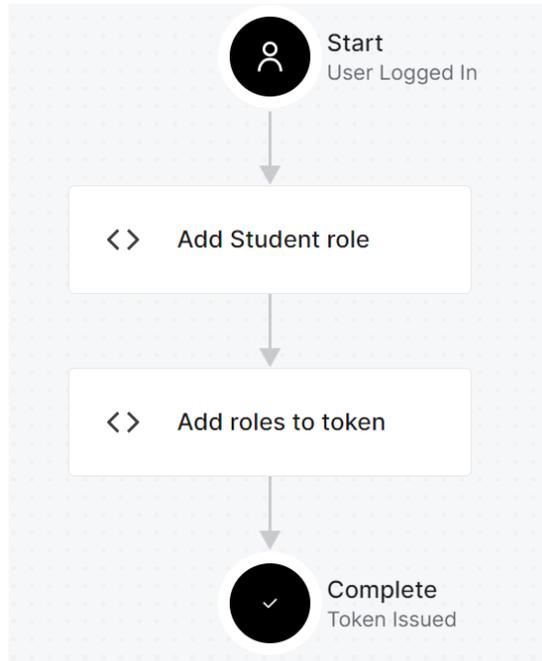
Deploy your action by clicking on **Deploy** button.

Use Actions

- Actions -> Flows menu
- Select **Login** option

- Now you can drag and drop your custom actions from the right hand side to the login flow in the following order:
 1. **Add student role** action
 2. **Add roles to token** action

After the setting you need see the following flow in the editor:



15. ábra Login actions flow

Login page

We have the **New Universal Login** page in the Skipper application, we can see that in the following picture.

16. ábra New Universal Login page

Select New Login Page

- Branding -> Universal Login menu
- Click on **Advanced Options** card in the bott of the page
- In the **Setting tab** you should select the New Universal Login experience

Advanced Options

Switch between login experiences or override Universal Login with custom HTML. [Show more >](#)

[Settings](#) [Login](#) [Password Reset](#) [Multi-Factor Authentication](#) [Custom Text](#)

Universal Login Experience

Switch between

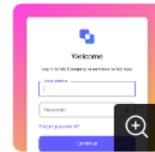
[New Universal Login and Classic Universal Login](#).

New Universal Login Experience must be selected when passkeys are used as an authentication method. [Learn more about passkeys](#).

New

Current implementation of Universal Login.

- ✓ Quicker load times than Classic
- ✓ No-code customization options



Classic

Previous implementation of Universal Login.

- ✓ Extensive custom code support



17. ábra How to select Login Page

Create Custom Login Page

- Branding -> Universal Login menu
- In this page you can customize the following properties as you wish:
 - Company Logo
 - Primary Color
 - Page Background Color
- In the **Customization Options** you can customize almost every element of the login page, like labels, buttons, fonts, etc..

Customization Options



▶ Try

Discard ▾

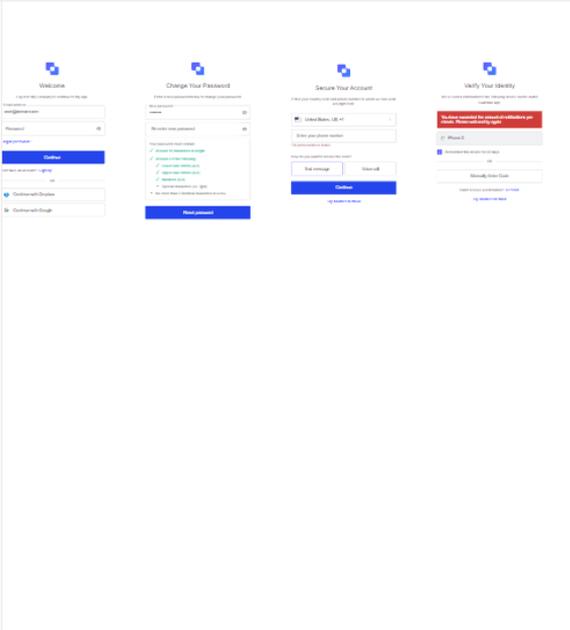
Save And Publish

All changes are live

Customize your login experience by selecting colors, fonts, and more.

Styles

- Colors
- Fonts
- Borders
- Widget
- Page background



Colors

- Primary button: #2445eb
- Primary button label: #ffffff
- Secondary button border: #c9cace
- Secondary button label: #1e212a
- Base Focus Color: #635dff
- Base Hover Color: [empty]

18. ábra Login Page Customization

Translations

The Universal Login page will be translated to the selected languages automatically, but you have to select the allowed languages in the Auth0 dashboard before that.

In the **Settings** menu there is a card with the title of **Languages**, this is where you can select the languages.

The screenshot shows the 'Languages' settings page in the Auth0 dashboard. It features a 'Default Language' dropdown menu currently set to 'English (en)'. Below this is a section titled 'Supported Languages' containing a grid of checkboxes for various languages. The 'English (en)' and 'Hungarian (hu)' checkboxes are checked, while all other checkboxes are unchecked.

Supported Languages	
<input type="checkbox"/> Basque (eu-ES)	<input type="checkbox"/> Bosnian (bs)
<input type="checkbox"/> Bulgarian (bg)	<input type="checkbox"/> Catalan (ca-ES)
<input type="checkbox"/> Chinese - Simplified (zh-CN)	<input type="checkbox"/> Chinese - Traditional (zh-TW)
<input type="checkbox"/> Croatian (hr)	<input type="checkbox"/> Czech (cs)
<input type="checkbox"/> Danish (da)	<input type="checkbox"/> Dutch (nl)
<input checked="" type="checkbox"/> English (en)	<input type="checkbox"/> Estonian (et)
<input type="checkbox"/> Finnish (fi)	<input type="checkbox"/> French - Canada (fr-CA)
<input type="checkbox"/> French (fr-FR)	<input type="checkbox"/> Galician (gl-ES)
<input type="checkbox"/> German (de)	<input type="checkbox"/> Greek (el)
<input type="checkbox"/> Hindi (hi)	<input checked="" type="checkbox"/> Hungarian (hu)
<input type="checkbox"/> Icelandic (is)	<input type="checkbox"/> Indonesian (id)
<input type="checkbox"/> Italian (it)	<input type="checkbox"/> Japanese (ja)
<input type="checkbox"/> Korean (ko)	<input type="checkbox"/> Latvian (lv)

19. ábra Select allowed languages

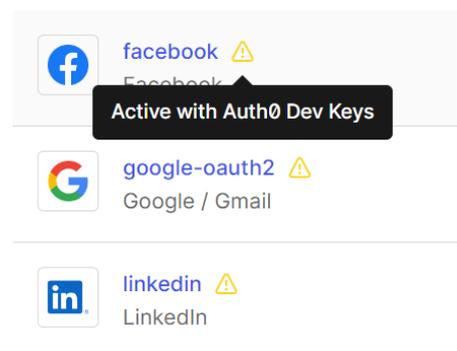
Social login settings

- Google
- Facebook
- LinkedIn

Social logins now works with developer keys in development mode. We need to set this to production with the following settings. If every setting is done, the warning icons will disappear from Auth0 "Authentication" > "Social" menu.

Social Connections

Configure social connections like Facebook, ¹



Google

Create a Google Cloud Project in the Google Cloud Console.

Create a new project if you don't have one already:

- Click on the project dropdown at the top, then "New Project."
- Name your project and choose a billing account if necessary.

Enable Google People API

- In your project, go to the "APIs & Services" > "Library."
- Search for "Google People API"
- Enable the "Google People API" API.

Create OAuth Credentials

- Go to "APIs & Services" > "Credentials."
- Click on "Create Credentials" and select "OAuth Client ID"
- Choose "Web application" as the application type.

Set up the consent screen if not already done:

- You'll need to fill in basic information, such as the application name, support email, and developer contact information.

- You might need to verify the app depending on its scope and usage.

← Client ID for Web application 

Name *

Skipper

The name of your OAuth 2.0 client. This name is only used to identify the client in the console and will not be shown to end users.



The domains of the URIs you add below will be automatically added to your [OAuth consent screen](#) as [authorized domains](#).

Configure the OAuth client:

- Authorized redirect URIs: Add the callback URLs that Auth0 will use, which you can find in the Auth0 dashboard on the top left corner.
 - `https://YOUR_DOMAIN/login/callback`
 - DOMAIN: <whole domain from your auth0 account (like in the picture)>.<Region>.auth0.com



Save and get the Client ID and Client Secret.

Additional information

Client ID	5570C
Creation date	August

Client secrets

If you are in the process of changing

Client secret	GOCS
---------------	------

Set up Auth0 with Google OAuth Credentials

- Go to the Auth0 Dashboard and navigate to "Authentication" > "Social."
- Select Google from the available connections.

In the General section enter your Google OAuth Client ID and Client Secret that you obtained from the Google Cloud Console.

General	Name
	<input type="text" value="google-oauth2"/>
	<small>If you are triggering a login manually, this is the identifier you would use on the connection parameter.</small>
	Client ID
	<input type="text" value="Leave blank to use Auth0 dev keys"/>
	How to obtain a Client ID?
	Client Secret
	<input type="text" value="Leave blank to use Auth0 dev keys"/>
	<small>For security purposes, we don't show your existing Client Secret.</small>

Save your settings.

Facebook

Create a Facebook App in the Facebook for Developers website.

- Log in with your Facebook account.
- Go to "My Apps" in the top right and click "Create App."
- In the " Use cases " tab you need to select " Authenticate and request data from users with Facebook Login "
- Fill in the basic details (e.g., App Name, Contact Email) and click "Create App ID."

After the app is created, you need to set redirect url for login. This can be done in the "Use cases " menupoint with clicking on "Customize" in the " Authentication and account creation " card.

Use cases

Add and customize main and available use cases on this app. Pick from permissions and other features to make your app work.

**Authentication and account creation**

Our most common use case. A secure, fast way for users to log into your app or game and for the app to ask for permissions to access their data to personalize their experience.

Added: Facebook Login, public_profile

[Customize](#)

In the left menu bar select "Settings" and add the " Valid OAuth Redirect URIs ". This needs to be the same as that you set in the GCP console.

Valid OAuth Redirect URIs

A manually specified redirect_uri used with Login on the web must exactly match one of the URIs listed here. This list is also used by the JavaScript SDK for in-app browsers that suppress popups. [?]

[Copy to clipboard](#)

Save your settings with clicking on "Save changes" button in the end of the page.

In the "App settings" > "Basic" menu, you can find the App ID and the App secret, that you need to set up in the Auth0 dashboard.

Set Up Auth0 with Facebook OAuth Credentials as you did for Google.

LinkedIn

Create a LinkedIn Application via the LinkedIn Developer Portal.

- Log in with your LinkedIn account.
- Click on "My Apps" and then "Create App."
- Fill in the required details (e.g., App Name, LinkedIn Page, Company, Email).
- Agree to the LinkedIn API Terms of Use and click "Create app."

Set Up LinkedIn OAuth

- After creating the app, you'll be directed to the application dashboard.
- In the "Auth" tab of your app dashboard, you'll find your Client ID and Client Secret.
- You'll need these for Auth0 configuration.

Configure OAuth Redirect URLs

- In the LinkedIn application dashboard, under "Auth", go to "Redirect URLs".
- Add the Authorized Redirect URLs provided by Auth0.
 - You can find these URLs in the Auth0 dashboard under the LinkedIn connection settings.
 - `https://YOUR_DOMAIN/login/callback`

Set Up Auth0 with LinkedIn OAuth Credentials