# User Authentication

Two options for authenticating the administrator users; custom-built authentication and authentication providers.

## Using providers

Using third-party auth providers like Auth0, Google, etc.

**Benefits:**

1. Handles all of the authentication logic and user credentials.
2. Easy to implement
3. Handles account management like resetting passwords.
4. Provides SDK for client-side authentication management
5. Allow the admins to log in with any account they want to

**Drawbacks:**

1. Third-party
2. Might cost money (unlikely in this case though as Auth0 allows 7000 active users for free)
3. Might need some sort of capabilities/interface to handle roles depending on client needs (most likely the case)

**Resources**

**API**

This [article](https://auth0.com/blog/securing-aspnet-minimal-webapis-with-auth0/) shows how to implement a minimal API in ASP.NET and set up authentication/authorization. The minimal API part isn’t applicable, but the other section is good.

**Client-side**

[Docs found here](https://auth0.com/docs/libraries/auth0-android) show how to use Auth0 client-side SDK with Android.

[Docs](https://auth0.com/docs/libraries/auth0-react) for using Auth0 client-side SDK for React SPA apps.

## Using custom solution

Creating a simple authentication solution using [OAuth2 standards](https://oauth.net/2/). It’s not too difficult to do, but it does take a bit more work.

**Benefits**

1. Don’t need to deal with any third-party providers making the solution simpler
2. Saving money (again unlikely this applies here)

**Drawbacks**

1. A lot more work (but also a good opportunity to learn more)
2. Possibility for security issues if not implemented correctly
3. Would have to handle token storing and refreshing on the client-side which again is more work
4. Create account management capabilities
5. Might need some sort of capabilities/interface to handle roles depending on client needs (most likely the case)

**Resources**

1. This [article](https://medium.com/@ajidejibola/authentication-and-authorization-in-net-6-with-jwt-and-asp-net-identity-2566e75851fe) walks through setting up the authorisation part but doesn’t include refresh token handling on API or client side.
2. This [article](https://jasonwatmore.com/post/2022/01/24/net-6-jwt-authentication-with-refresh-tokens-tutorial-with-example-api) covers the whole implementation.
3. More information on how the whole flow works are found [here](https://oauth.net/2/).

**Summary**

Using a third-party authorisation provider is definitely easier, all of the account management functionality is handled, and it even has client-side SDKs to manage the authentication state on client apps but this complicates the project by having another piece of the system to worry about, not necessarily for us, but for the client.

The custom solution route contains more work, we would have to be a lot more careful with the implementation, but there are plenty of resources only to help with this. We would also have to implement the client-side authentication logic as well as the account management logic, including email sending when resetting passwords. With that being said, this is a good opportunity to learn more about authentication, HTTP and many security factors.

The obvious winner is using third-party authentication.

**Some things to discuss and decide together**

1. Allowing anyone to sign up / invite link or somebody just creates accounts via the Auth0 dashboard?
2. Does the client have enough technical knowledge to assign and removes roles as needed?

## Project Authentication

API keys can be used to authenticate different projects (drawing apps). Multiple authentication schemes will have to be set up and then specify which one is used for each controller.

To implement this we can just create a simple key and store that in the API’s config file, then create a middleware that checks request headers for that key. Store the same key in the drawing application config/.env files and use it throughout the app when making requests.

The only issue is that changing the keys will require some technical knowledge but I don’t think there is much we can do about that.

**Resources**

1. [Using API Key authorization](https://swagger.io/docs/specification/authentication/api-keys/)
2. [Implementing API Key authentication](https://www.c-sharpcorner.com/article/using-api-key-authentication-to-secure-asp-net-core-web-api/)
3. [API Keys for project authentication](https://cloud.google.com/endpoints/docs/openapi/when-why-api-key#:~:text=API%20keys%20are%20for%20projects%2C%20authentication%20is%20for%20users,-Cloud%20Endpoints%20handles&text=API%20keys%20identify%20the%20calling,using%20the%20app%20or%20site.)