Authentication and Authorisation

Watched videos in breakout rooms and had to answer the below questions.

Resources

- Authentication: https://www.youtube.com/watch?v=tZPENgB_Rd0
- Authorisation: https://www.youtube.com/watch?v=gnZo3BJgjpI
- Authentication vs authorisation explained: https://www.youtube.com/watch?v=2iTTBJd46ow
- Authentication vs authorisation: https://www.youtube.com/watch?v=85BBfKo6bdo

Questions to answer

- What is authentication?

Authentication is the process of proving that you are who you say you are, and gaining access to a product or service based on your identity (e.g. Are you who you say you are? could be credentials, biometrics etc.)

- What is authorisation?

Authorisation refers to the process of verifying what a user has access to (e.g. Do you have permission to do what you're trying to do?)

- When might you want to add authentication/authorisation to your app? When might you not want to?

Do: Banks/Financial Services, anything where data is persisting, where access needs to be paid for, most CRUD routes apart from READ (but include READ when info is sensitive), to avoid overloading a server
Don't: Where just GET requests are being made to a public data source (providing this won't overload the server, otherwise may need it), where we don't need to protect the data

While often used interchangeably, Authentication and Authorisation represent fundamentally different functions. In simple terms, Authentication is the process of verifying who a user is, while Authorisation is the process of verifying what they have access to. Real world comparison is authentication is your passport (to verify who you are) and authorisation is your boarding pass (Which plane you are actually allowed on/Which destination you are allowed to travel on)

Authentication helps keep your API secure as it ensures people can't be anonymous.
Authorisation helps define access to certain parts of your API – could be based on roles (e.g. Accountancy software has payment administrator, company owner etc so only certain people can make payments).
Authentication tends to happen once per session, authorisation should happen each time the user accesses the resource

We could build it ourselves, but there are plenty of third-party solutions which may be sensible to use because they will have spent many hours on solving the particular problem. Auth0 is an industry standard way of implementing authentication/authorisation. Amazon Cognito is also common.

Remember there are always trade-offs! Don't always take resources at face value, they may not be the gospel truth!

More videos and questions to answer:

Resources:

- https://withblue.ink/2020/04/08/stop-writing-your-own-user-authentication-code.html
- https://www.youtube.com/watch?v=Hh_kiZTTBr0
- https://auth0.com/blog/5-massive-benefits-of-identity-as-a-service-for-developers/

Questions to answer:

- What are the pros/cons of implementing it yourself?
- What are the pros/cons of using a 3rd party service to manage it for you?

My takeaways:

These resources seemed skewed towards the benefits of third-party implementation. The pros of third-party seem like the cons of implementing it yourself and vice-versa.

Pros of third-party implementation:

- Can be faster to implement (they've set it up, you just call the API)
- Can save your valuable time (might be cheaper than the hours you will put in)
- Can be more secure (as they're security experts) as it can be easy to get things wrong if building yourself
- No need to store your users' data yourself
- User data could be more secure as bigger companies can be more robust against hacking
- Like an "insurance" against liability (if you got compromised and lost data)
- Third parties often support multi-factor authentication (MFA) or security certificates/keys. MFA can prevent up to 99.9% of account compromise attacks according to Microsoft report
- Because of large scale of number of sign-ins, third party companies can better identify suspicious patterns (often using AI)
- Third-party companies will keep up to date with the latest types of attacks/hacks

Cons:

- Not always more secure (can you trust private companies?) than open source (e.g. Passport.js for node)
- Can't store the data yourself (giving your user data away)
- Price (Auth0 can be expensive – you need to ensure it doesn't eat away your revenue)
- Lack of flexibility – you are constrained by the limitations the third-party company puts in place
- How do you test third-party implementation is completely secure?
- Using third-party solutions don't help you learn about what's actually happening (Back-end developers should learn how to set up authentication themselves)

Other options for authentication/authorisation:
ORY Hydra
Keycloak
BeCrypt

We are going to focus on integrating with third-party implementations as that is what is likely to happen in a company (although it would be useful to learn what's going on under the hood!).

Then sent down in our pairs to follow a tutorial to integrate Auth0 in to a workshop repo, using the first 100 seconds of this video: https://www.youtube.com/watch?v=yufqeJLP1rI.