Authentication & Authorization

SOFTWARE SECURITY

- Software security should always be a high priority for product developers and their users
  - If you don’t prioritize security, customers will inevitably suffer lossed from malicious attacks
  - Worst case scenario, product providers out of business
  - Recover from the attacks take time and effort

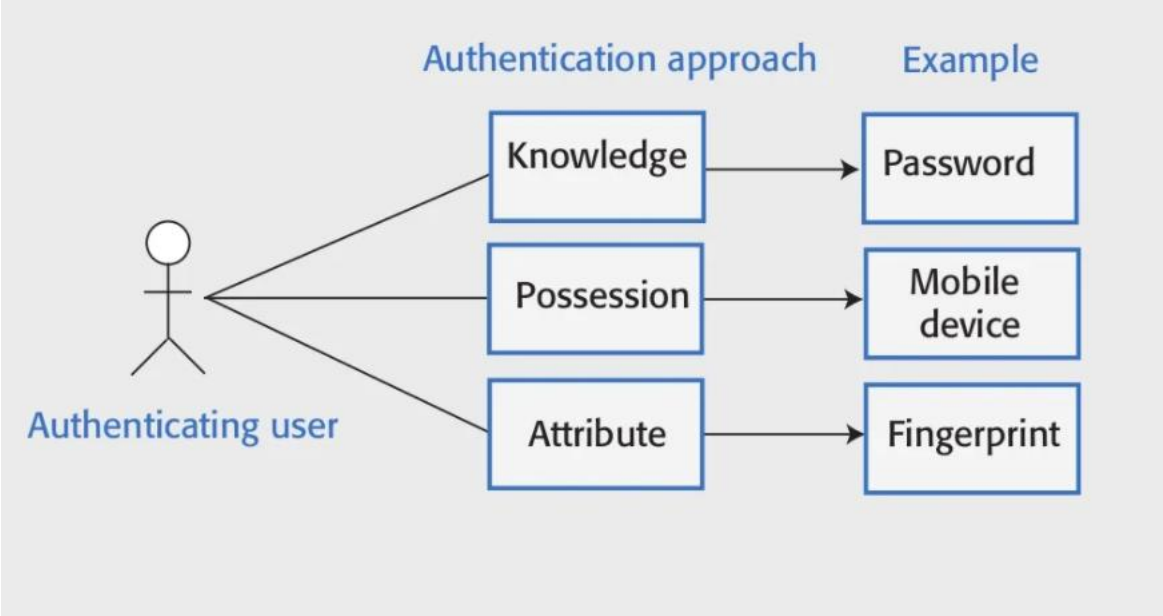
- *Examples: unauthenticated users* | *unauthorized users*

AUTHENTICATION

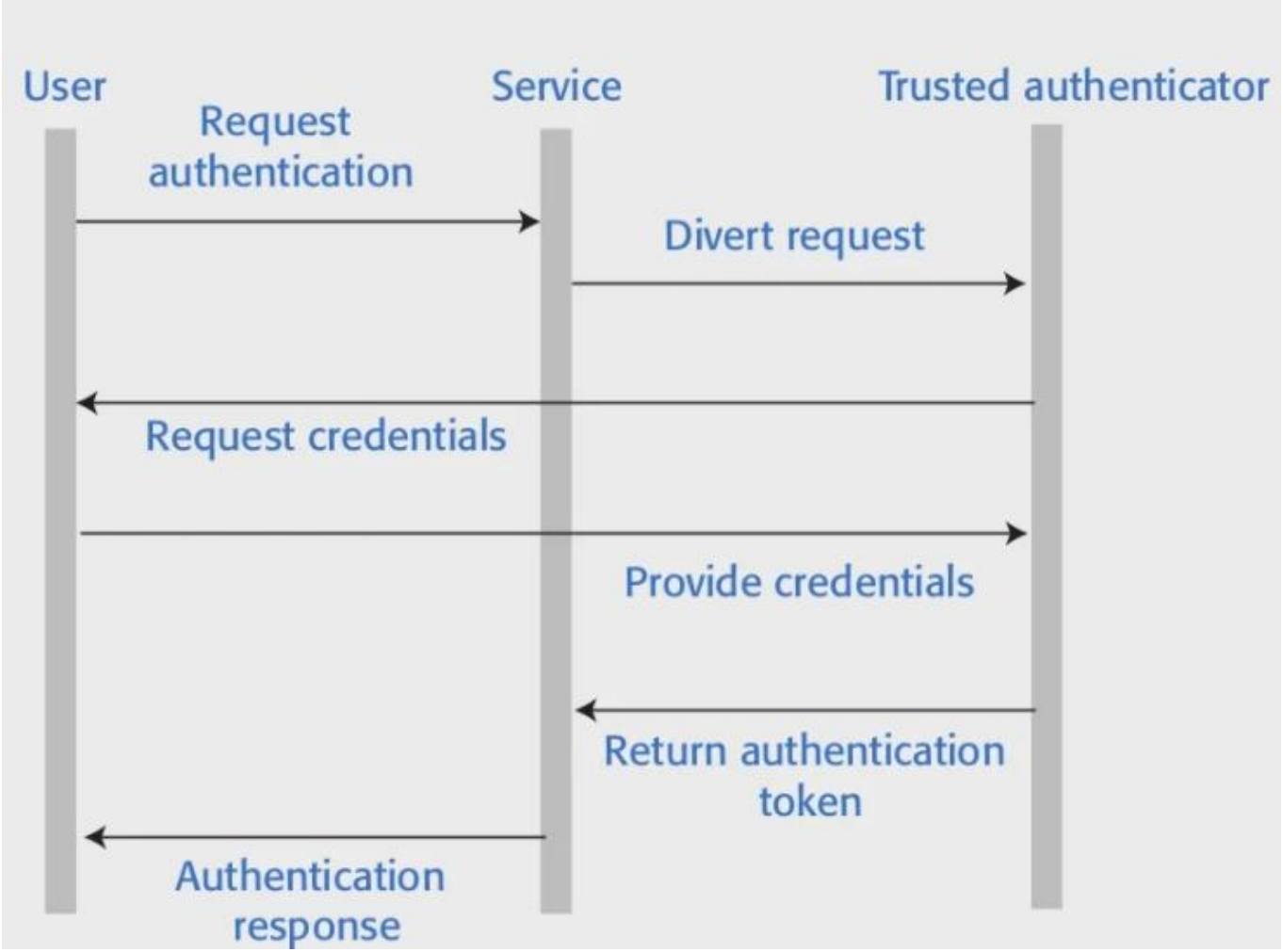
- Authentication is the process of ensuring that a user of your system is who they claim to be
- Authentication is needed in all software products that maintain user information, so that only the providers of that information and access and change it
- Also, use authentication to learn about users. Hence, personalize their experience of using the products

AUTHENTICATION APPROACH

- Knowledge-based:
  - The user provides secret, personal info to register. On logging in, the system asks them for this info
- Possession-based:
  - This relies on the user having a physical devices that can generate or display info that is known to the authenticating system. The user input is info to confirm the authentication
- Attribute-based:
  - Base on a unique biometric attribute of the user such as fingerprint
- Multi-factor



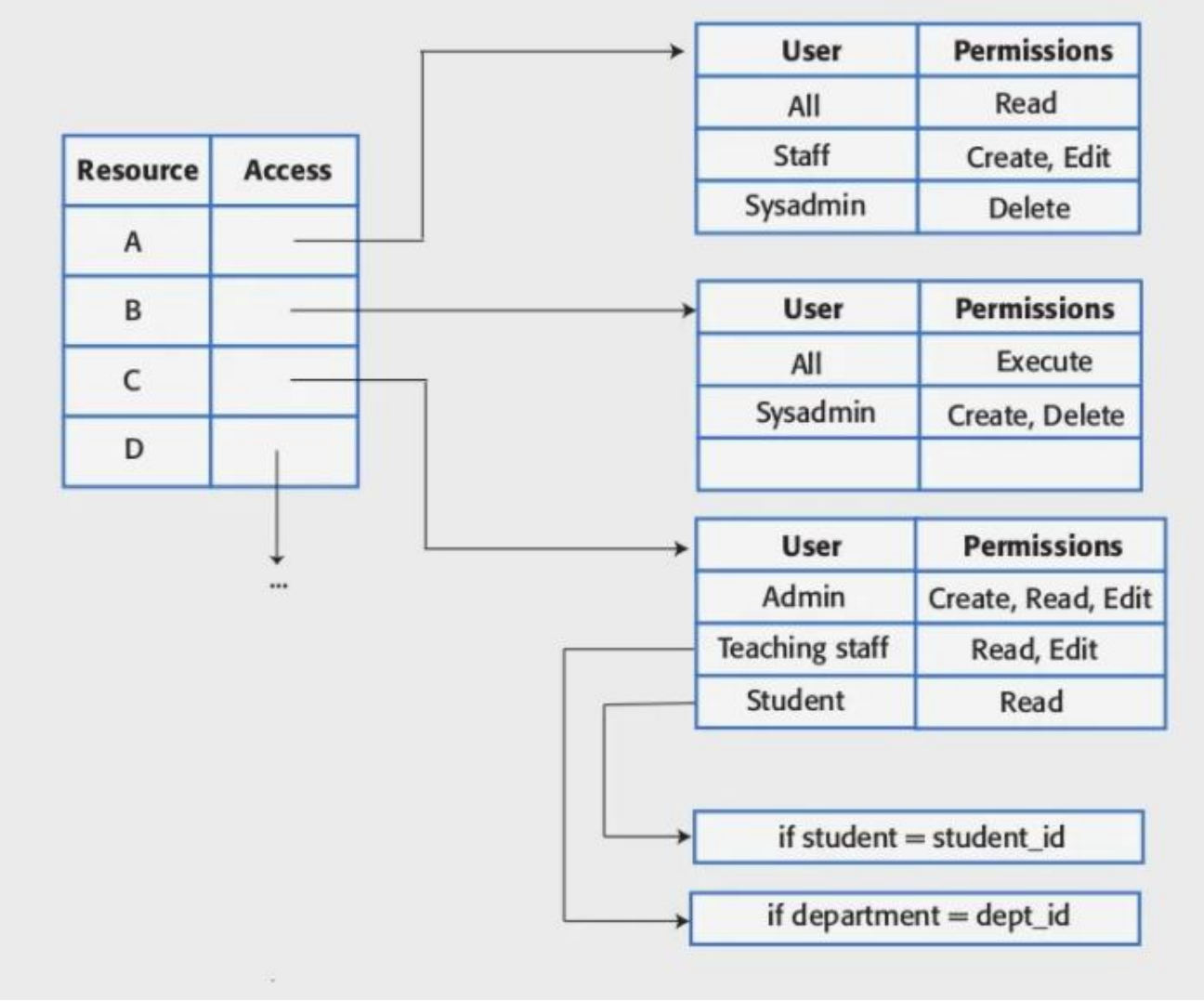
Authentication using an external authentication service



AUTHORIZATION

- Authorization is a complementary process in which that identity is used to control access to software system resources
  - for example: a doc on Google Drive, the document's owner may authorize you to read the content but not to edit the content
- Authorization is based on an access control policy

Access control lists

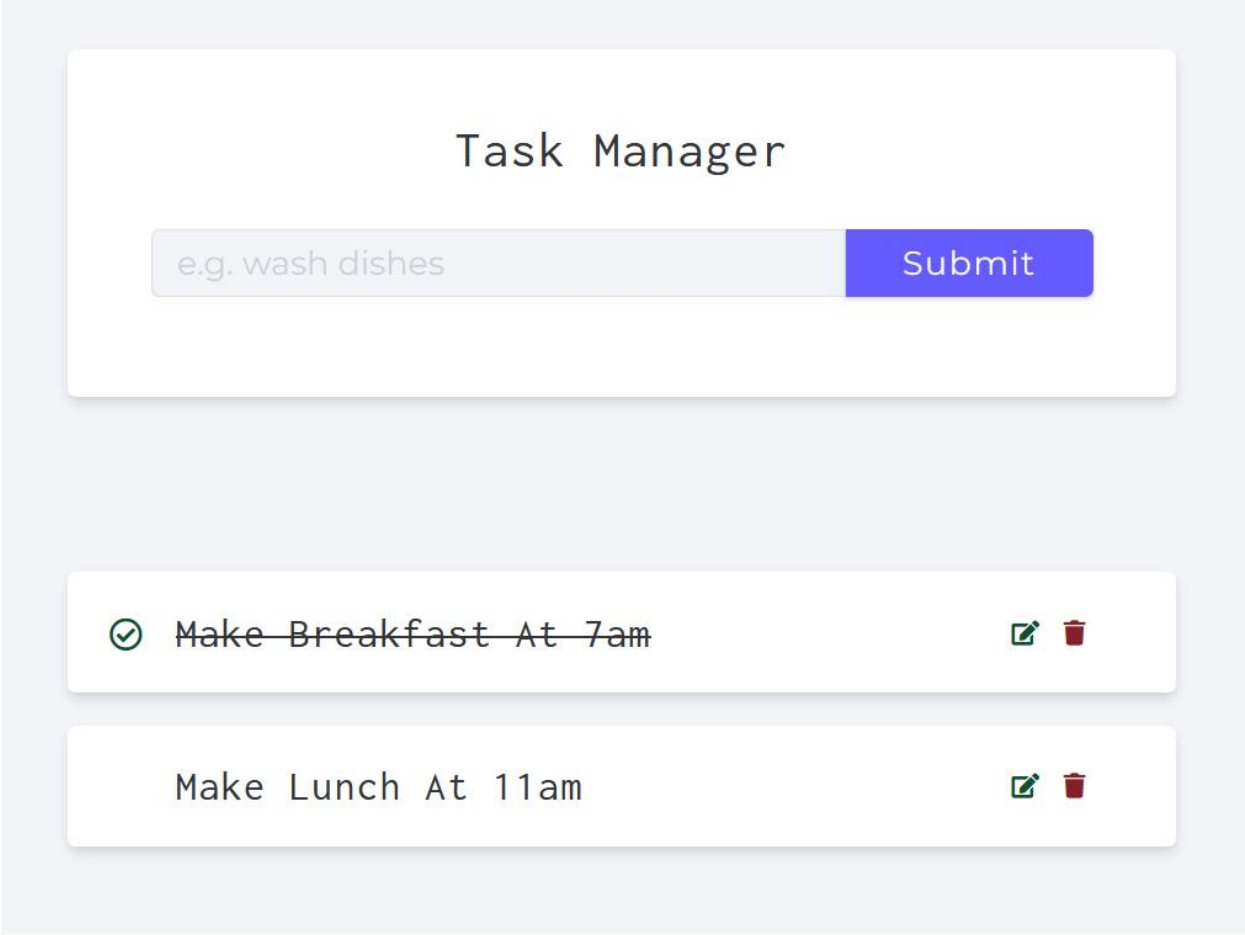


TUTORIAL

Using *bcrypt* and jwt to create login/register API

TASK MANAGER

- Functionality: Manage tasks for a specific user, who is able to register for an account.
- The user should provide at least two kinds of information: username and password
- Authenticate users by their encrypted password
- Authorize users to accsess only their tasks.



INSTALLATION

- *npm install bcryptjs*
- *npm install jsonwebtoken*

PASSWORD ENCRYPTED

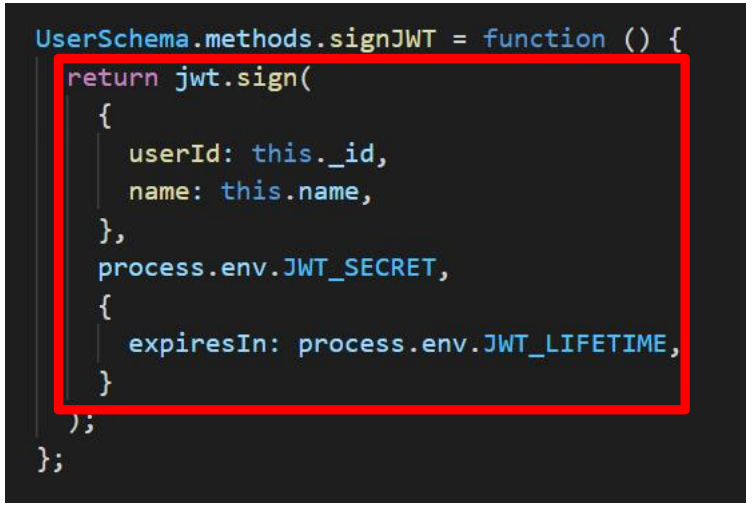
```
UserSchema.pre('save', async function(next){
    const salt = await bcrypt.genSalt(10)
    this.password = await bcrypt.hash(this.password,salt)
    next()
})

UserSchema.methods.comparePass = function(candidatePass){
    const isMatch = bcrypt.compare(candidatePass,this.password)
    return isMatch
}
```

jsonwebtoken

- Authorize user to access specific APIs by token
- Once the user is logged in, each subsequent request will include the JWT, allowing the user to access routes, services, and resources that are permitted with that token

GENERATE TOKEN

- SECRET KEY
- Sign with *jwt* to generate token
- Transmitting data between client and server with the generated token



TEST YOUR APIs

GO FURTHER

- https://www.npmjs.com/package/bcryptjs
- https://jwt.io/introduction