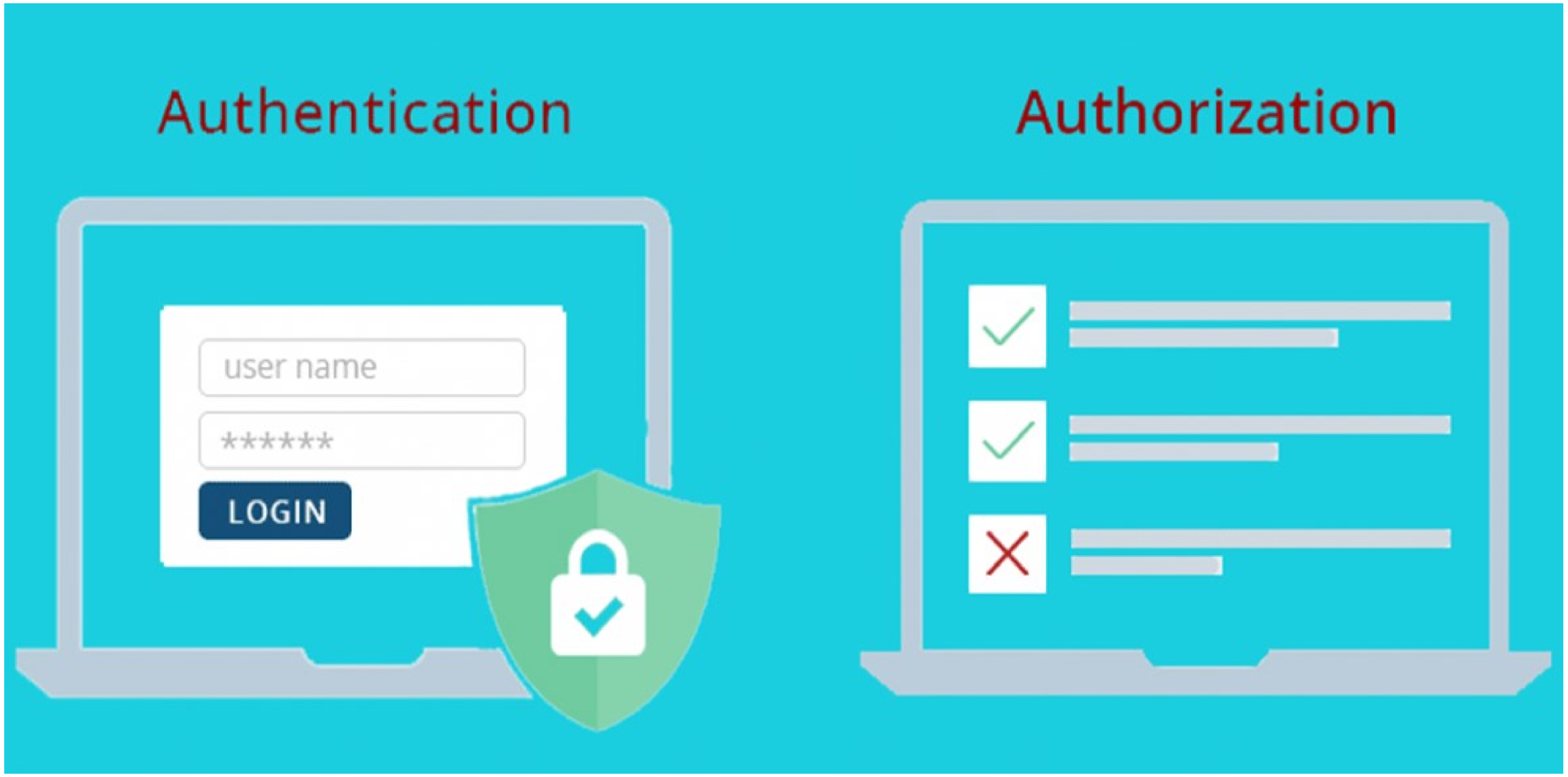
HTTP Session Management

- **Authentication** is validating that an entity is who or what it claims to be.
- **Authorization** is validating that a user can perform a given action



What is a Session?

- HTTP is a **stateless** protocol
- So, we need to have some logic to keep track of the previous requests to the server
- Session is a server-side storage of information to persist throughout the user's interaction to the web application
- A **unique identifier** is stored on the client side (session id)
- This identifier is passed on every request to the server
- This identifier is matched by the server and retrieves the information attached with the id

Sessions in Node.js

- Sessions in Node.js are stored using 2 ways:
  - Session state Providers:
    - Cookie + backend store
  - Default sessions:
    - client-sessions or express-sessions module

Sessions in Node.js

- Client-sessions npm module provides simple implementation npm install client-sessions
- These sessions are limited to application scope
- So when the application is restarted, these sessions are invalidated

```
//Include default session
var session = require(‘client-sessions');
```

Sessions in Node.js

```
var session = require('client-sessions');

app.use(session({
        cookieName: 'session',
        secret: 'cmpe273_test_string',
        duration: 30 * 60 * 1000,
        activeDuration: 5 * 60 * 1000,
      }));
```

Sessions in Node.js

```
var session = require('client-sessions');

app.use(session({
        cookieName: 'session’,            //cookie-name stored on browser

        secret: 'cmpe273_test_string', //secret_id stored
        duration: 30 * 60 * 1000,  //how long the session will stay valid in ms

        activeDuration: 5 * 60 * 1000,    //if expiresIn < activeDuration, the
                                          session will  be extended by
                                          activeDuration milliseconds
        }));
```

Sessions

- We can use the create a session using request object

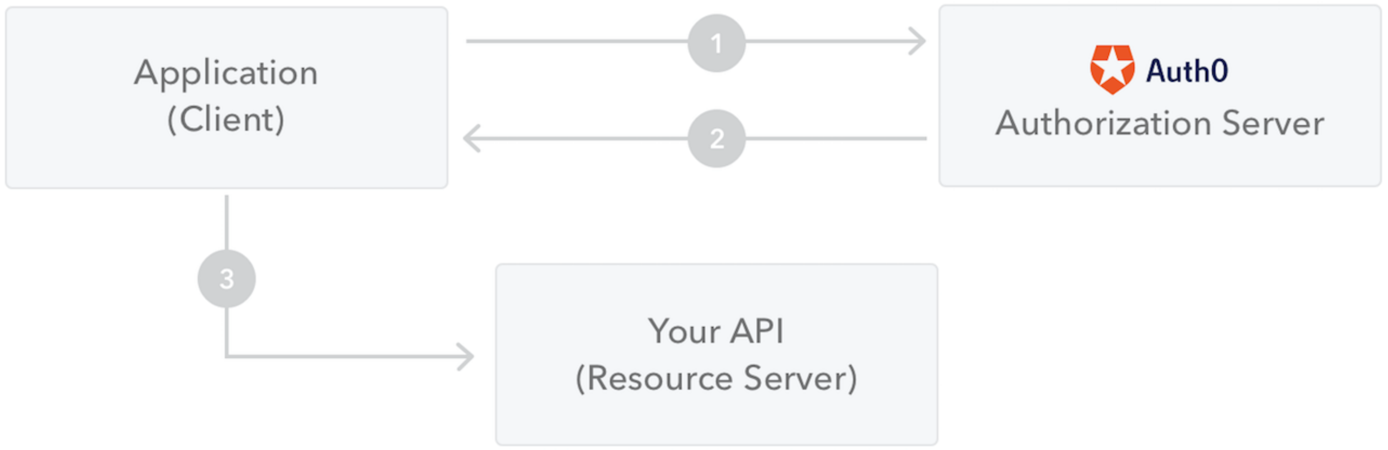
```
//store the username and email
address after successful login
req.session.username = username;
req.session.email_address = email_address;
```

Passportjs

- Passportjs is capable of performing authentication and storing sessions
- Stores the sessions on external store – in MySQL database
- Every time a request is sent, after the session is created, we can check whether the session exists
- If session doesn't exist, redirect user to the login page
- ***Independent of server which receives the request, as the session is stored on the database***

JWT (JSON Web Token)

- The application or client requests authorization to the authorization server. This is performed through one of the different authorization flows.
- When the authorization is granted, the authorization server returns an access token to the application.
- The application uses the access token to access a protected resource (like an API).


Application
(Client)
1
2
Auth0
Authorization Server
3
Your API
(Resource Server)