Authentication Steps:

First we will register users, then provide login for that user.

=> **Lodash package:** it is used to pick only selected properties of object. For example we want to pick only name and email of user and not the password. **We use \_.pick() method for that**

syntex: \_.pick( <object\_name >, [array of elements] )

e.g 1: \_.pick( user, [‘name’, ‘email’] )

e.g 2: \_.pick( req.body, [‘name’, ‘email’, ‘password’] )

# Hashing password (bcrypt password):

1) bcryping (hashing) password:

=> **Brcypt package:** this package used to hash the password. Example we should store hashed password in database instead of plain password, for safety. **Where we first generate a salt then use that salt to hash the password.**

Syntax: let salt = await bcrypt.genSalt(10);

Let hashedPassword = await bcrypt.hash(‘password’, salt) ;

E.g: let salt = await bcrypt.genSalt(10);

let user.password = await bcrypt.hash(user.password, salt) ;

await user.save();

2) de - bcryping (rehash)password (checking password is correct ? ):

=> **Brcypt package:** We will use compare() method of bcrypt to check users entered password is same as the password stored in DB or not, **if it is same then user will login successfully.**

**Synatex & Example:** var validPassword = await bcrypt.**compare(**req.body.password, user.password**)**

# JWT (JSON web token):

**How its work JWT:** it will act as an id card for the request, below steps for explaination

**Steps:**

1. first time user login then server will send JWT to browser and browser will store this jwt.
2. The next time when same user generate any request to server, it will show that JWT token as id card.
3. If the JWT is correct then server will serve the request, if it is not correct then will give authentication error.

=> JWT: it is consist of 3 – parts 1) Header, 2) Payload, 3) digital signature.

**1) Header:** this will contain 2 properties i.e: “alg”, “type”. Where alg – algorithm used in jwt, and type – jwt.

**2) Payload:** this will contain the properties of the user like name, email etc.

**3) Singature:** if anyone changes JWT or make any edit to payload then signature will fail, because signature is used the private key that will only available on server. And hackers don’t know about that.

**jsonwebtoken package:** we use this package to generat jwt token that we will send back as response for successful user login. We can import that apackage as jwt const.. **We use .sign() method of jwt package in whick first argument will be the payload then second argument will be the private key.**

**Syntex & Example:**

const jwt = require(‘jsonwebtoken’);

const token = jwt.sign( { \_id: user.\_id, name: user.name }, ‘jwtPrivatekey’ ) ;

res.send( token ) ;

***COVER Till …*** 5\_vidly8.9-Auth-jwtTokenGeneratedInLogin ***File.***