00:00:00 - 00:01:08

multiactor authentication is a security method that requires user to provide at least two different forms of identification to access a system in this video we are going to build a two-factor authentication system using nodejs Express Speak Easy and passportjs in this project we will learn how to secure user accounts with 2fa improving the overall security of our application so if this sounds interesting then stick around also don't forget to subscribe the channel and press the Bell icon so

00:00:35 - 00:01:37

that you don't miss the videos like this one so let's get [Music] started all right guys so before we jump to the visual studio code and start building our project we will just go through some slides and understand how our project structure will look like and what is two- Factor authentication that we are building here and what are the different routes that we have to build so we will first start with our application structure so this is how our application structure would look like we have a user client and this user client

00:01:07 - 00:02:02

is going to send a request and get a response from our server and this will be an Express server where we will be having our app which will be the main entry point of our application now this app will take the configs which are the passport JS config which we are going to use it for the authentication purpose we will come to that later on and then there will be the database config where we will be establishing the connection of database with mongodb now this app will also contain all our routes so we

00:01:35 - 00:02:29

are going to have the Au routes and the multiactor authentication routes and these routes will have all the logic into their controller which will be having a model which we are going to create the user model with the help of mongus schema and then we are going to store the data in the mongodb database so this is a rough idea how our application is going to be structured now we are going to take a larger look on the routes that what are the routes that we have to build so here you will see that we have the routes and we are

00:02:02 - 00:03:05

going to have the four Au routes which will be the register a user login a user log out a user and check the status of a user whether a user is authenticated or not authenticated and for this login part we are going to use the passport JS and we are going to use the strategy for the passportjs as local strategy so when we will implement it we will see what is passportjs and how we are going to integrate it into our project to handle the Authentication part for setting up of the multiactor authentication we are

00:02:33 - 00:03:36

going to have three routes which will be the setup we are going to set up the multiactor authentication into a project then we will verify the token and the third one is if we want to reset our uh MFA then we can also reset it and then set it again and for the setup and the verify we are going to use a Speak Easy which is a node module which will help us to generate the time based OTP so we will see what is this dependency is and uh how we use it into the project to set up MFA then let's understand how this

00:03:04 - 00:04:02

multiactor authentication uh looks like so here you will see that we are going to have our user which is going to log in using the username and the password and when the username and the password matches that will be the first Factor authentication once the username and the password match we will need a another authentication which will be the second Factor authentication or we say MFA multiactor authentication that will have a authenticator app on the users's device and there will be a time based

00:03:33 - 00:04:30

OTP which is going to be sent so whenever we have this time based OTP we will need the OTP also to verify the user so that means to announce the security of the user we are having these two steps in order to user get authenticated and then access our resources on the server so this is how in a nutshell the multiactor authentication is and we are going to build uh the project uh we are going to start for the backend project first build the backend apis so I will go to the visual studio code and in the visual

00:04:02 - 00:05:06

studio code you will see that I have an empty folder which is the react nodejs multiactor project now the first part what we have to do is we go to the terminal so I will just open my terminal and inside the terminal we are going to init our uh package Jon file so let's have npm init hyphen Y and this is

going to initialize our package Json file so I'm going to hit enter and yes we have the package Json file now in the last project in the role BAS Access Control we were using the common JS to import

00:04:34 - 00:05:29

now in this case we are going to use the module es6 module so what we are going to do is uh I'm going to use the type here and I'm going to use the type as module so that we can use the import statement instead of require I will have the module here okay in the scripts we need to add the scripts as well uh I will have the dev here and then I'm going to add a script here right now we don't have any script here now we will start our project by installing all the dep dependencies that we need for this

00:05:02 - 00:06:07

project so let's do that one by one so I will clear the console now I'm going to do npm install the first we need to install is the express so we are going to have the express I will use npmi to install it now after Express we will need the EnV so that we can uh fetch the values from ourv file so I will have the EnV then we will also need to have the passport so I will use the passport we will go to the passport. JS official Doc and we will see that what what the passport.js is and what capability it

00:05:34 - 00:06:30

provides with the passport we will also need the passport local so I will have the passport local and this is uh basically the local strategy which we are going to see again then we will also need to have the course we are going to use the backend apis once we build the backend project we are going to make use of these apis and call them from our frontend react application so that's why we will need the course so that it will enable the cross origin request then we will also need the bcrypt so that

00:06:01 - 00:07:03

whenever we register a user we can hash the password so we will not store the raw password so let's have the bcrypt JS then we will also need a jwd token so that we can uh pass the Json web token to the user once the user is authenticated so I will have the Json web token we will also need the mongus so I will use the mongus we will also need the Speak Easy so I will also show what the speak easyy is about and then we will need a QR code so I will use the QR code library and we will need the

00:06:32 - 00:07:36

mongos so these all are the dependencies that we are going to require for this project let's go one by one so we have the express we all know Express is used uh as a framework for nodejs to build the rest API we have a EnV to fetch the values from ourv file passport and passport local we skip for now course we know that we use course in order to allow our apis to be accessible from a different origin bcrypt JS we have used it couple of times in the past we need this to Hash our raw password Json web

00:07:05 - 00:08:05

token in order to handle the authentication part we will provide the token to the user Speak Easy I'm going to talk it about uh later on when we talk about the passport JS QR code it's a simple Library which is going to generate a base 64 string whenever you pass some data to it so it's going to generate a base 64 string which you can just decode it and get a QR code then mongus we use the mongus just for the user schema and and then uh have our card operation into mongodb one more dependency we will need that is the

00:07:35 - 00:08:32

express session because we are going to use the passportjs and passport local so it works very well whenever we need to create or establish a session so let's have the express session which is going to generate a session ID for us whenever we have the uh user authentication established so I'm going to hit enter and I will first let all the dependencies get installed so I'm going to hit enter all right so now we see that all the dependencies have been installed here uh now we will also use

00:08:03 - 00:09:05

the node Monon so I will have the npm install node Monon hyphen hyphen save hyphen Dev I want it as a Dev dependency so here I will have the uh node modon so the nodemon is also installed as a Dev dependency now we are having this index.js which is our main entry point so what I will do is uh I will go and create the folder structure so I will go here create a source so this is the source along with that we also create a EnV file which will hold our environment variables now in the source file I'm

00:08:35 - 00:09:35

going to create the index.js all right and now since we have the index.js here uh we can create our simple Express server but before we go and do our Express server I want to give you a bit of idea

about the passportjs and the speak EZ so let's go to the browser and let's check so if I go to the browser you will see that this is the passport.org it's a simple middleware which provides authentication for the nodejs and and you can use different strategies they have around 500 plus strategies so if I click on

00:09:05 - 00:09:56

view all strategies then you will see that there are so many strategies if you want to integrate it with the Facebook authentication then you can just use passport Facebook for us we are going to use passport hyphen local uh because we are just going to use the username and the password in order to authenticate the user so this is how we are going to use it we have already installed the passport local and it's a middleware so this is what we are going to make use of passport. use and and there will be our

00:09:30 - 00:10:31

strategy so we will just make use of it when we are actually writing our application and the another package which we are using is uh I will just go to the npm and Speak Easy so this is a Speak Easy it provides a twoof factor authentication for our nodejs application so Speak Easy it is the onetime password generator and it is basically used if you want to include MFA into your project it also supports the Google Authenticator or any other two-factor authentication device so there are a couple of authenticator apps

00:10:01 - 00:10:57

like Google Authenticator Microsoft authenticator AI so any of the authenticator which you have it on your device you can just use it and Speak Easy will work with them so the simple way we need to install it and then in order to use it what we need to do is it will first help us to generate a secret and then uh that secret will be shown in the form of a QR code which a user can scan from their authenticator app and then when the authenticator app reads the QR code it is going to show you a token which is a Time based but you

00:10:29 - 00:11:33

don't have to worry about it we are going to implement each and every step in a manner which will make you to understand the two Factor authentication very easy so for now you just need to understand that Speak Easy is an npm package that we are using to generate our two Factor authentication token and everything and passport is a library that we are going to use in order to

have authentication in our project and passport local is our authentication strategy because we are going to use the username and the password in order to

00:11:00 - 00:12:20

authenticate the user maybe let's go back to our Visual Studio code all right now we have this index.js and in the index.js we can simply start by creating a simple express application so I will go here I will have the import Express from Express then I will also import the session from Express session then I will import the passport then I will also import the EnV so let's have the Dov from Dov then I will import the course so let's have the course from course all right I missed the codes here

00:11:40 - 00:12:50

so I will just put the codes here so we have imported the dependencies that we need uh now let's create a simple uh Express app so before that I will just use the env. config so that it can fetch the values from ourv file now let's create the constant app and we will have our Express app app now on this Express app what we can do is we can attach the middlewares that we need so I will have the middle wees and the first middleware we need is the app. use and we are going to have the express Json so I will have

00:12:15 - 00:13:28

the Json which is coming from the express and here we can also pass the limit so I want to limit my data transfer which is should be 100 MB of Json data all right then I will also make use of app. use and URL encoder so we will have the URL encoder and here also we can have the limit as 100 MB and we can have extend as truth and we are using these two middle beares in order to know our express application that we can get an incoming request as in the form of a Json or it can be in the form of a URL encoded payloads all right so

00:12:51 - 00:13:56

we have these two middlewares now we will have a routes here for now I will just use it as routes and I'm not going to provide any routes now after the routes we will just listen the app so I will have the listen our app so let's create a constant port and we are going to use the port from ourv so let's give a port here so I will have the port as 71 all right so now we can go back here and we can just take the port value as process. env. port and if this value is not available then we can assign 7002

00:13:24 - 00:14:26

once we have this we can just listen to our app so app. listen is a method and we provide the port and this also gives us a call back so we can have a call back here and let's have console. log we can use the back text and we can have server is running on Port and we can just Define our Port so this is way we know that our server is running on the port so I will have the port all right now we want to give a try if we want to check the server is running on the port 7001 or not what we can do is uh we can

00:13:55 - 00:14:56

go to the package Json file and here we can have the node Monon so index.js so I will just change this to source and now we can give a try so what I will do is I will have npm run Dev and we see that the server is running on the port 7001 all right so we have created a simple Express server here all right so now next what we will do is we are going to use the course as well so let's have here middleware I'm going to create a constant I will have the course option and this course option will be an object

00:14:25 - 00:15:29

then inside this I'm going to use origin and here I can just Define all my Origins that I want to allow so for now I will just use HTTP col localhost 3001 and I assume that our react application will run on Locos 30011 if you have a different port you can just put it in the origin now along with that I will also use the credentials as true so whenever we have this cross origin we will also need to have this credentials as true with the help of this credentials true it will allow us that whenever we get a request from 30001 it

00:14:57 - 00:15:59

is also going to forward us the cookie which are the part of credentials so that's why we have this credential as true okay now we can make use of it so what I will do is I will have the app use and we can just use the course here and then we can provide our course option so I will have the course option okay now we will establish the session as well so I will have the app. use and in use we are going to have the session so this is basically the express session so if we go to our uh npm and if we

00:15:28 - 00:16:21

search for the Express session then we will be able to understand more about this package so here you can see everything about the session uh it will allow us to create the session and then you can

also provide some options whenever a session is established a session ID is created and the session data is stored on the server side I will show you once we create the session and then when we have the apis I will show you in the cookies how we have the session ID all right so we are going to have the session ID and then here we can

00:15:55 - 00:16:56

provide some options so for that what I will do is uh first uh in the session you always have a session secret which helps you to identify the established session so what I will do is I will have the secret here and here we have to give some meaningful secret name so what I will do is I will go here and I can have a session secret and this I can just equals to maybe my session secret but when you create a real project it should be some meaningful name so that any hacker or anyone cannot just decode your

00:16:26 - 00:17:26

human readable session secret and then just intercept your request so here what I will do is I'm can fetch this process. env. session secret and if there is no secret then I will just simply use the secret all right so we have this and then along with this we need to provide two more properties so one you will also see it here the undefined save option we have to define the resave option so let's have the resave and we have to provide the resave as false and basically this resave are whenever you create a session

00:16:56 - 00:17:55

the session data is stored in the local Express memory and there I have one more property which is uh save uninitialized so that also I have to make it false so you will see that if you h on it you will can read it that forces is a session that is uninitialized to be saved to the store and here along with this you can also provide more options like we are generating cookies and having a session ID so we can also provide the max age uh we want it to be stored on the user browser so I will have maybe I will store it at for 1 hour

00:17:25 - 00:18:26

so we can use 60,000 and If I multiply it by 60 then it becomes 1 hour all right so we have added the session part so now this is done uh we will also need to initialize our passport for the project so for that I will just go here and I will use app.use and we will have the passport. initialize so this is going to initialize the passport into our project and then uh what we need is we will also need to use the session so we have passport. session since we are using the express session so we will be using

00:17:56 - 00:18:51

passport. session so this is how we initialize the passport.js into our project all right so we are done with our index.js now what we can do is we can just create our folder structure for our project so first what I will do is uh I'm going to create the routes folder which is going to contain the routes after the routes I'm going to create the controllers which will contain all the logic of our routes then we are going to create the models which is going to contain our user model and last we will

00:18:24 - 00:19:31

need the configs which will have our passport config and the database connection so these are the fers that we will required for our project uh we can just start by connecting the database first so I will go here and I will create a file as DBC connect. JS oops it should be DBC connect. JS and to have the database connection established uh what we will need to do is import connect and this connect will comeing from the mongus then I can simply create a constant DB connect and this is going to be equals to an arrow function all

00:18:57 - 00:19:56

right and then we can just export it so I will have the export default DB connect okay now we're going to use the async so I will have the async here and let's write the connection to the database so uh I'm going to have a try catch block so that if there is an error we can catch the error all right and in the error what I will do is I will have the console. log and and here I will have database connection failed and we can just log our error whatever the error we get so I will have the error

00:19:27 - 00:20:25

okay and if there is an error we will use the process exit and we will exit the program so I will have process exit one now in the triy part what we will do is let's have the mongod DV mongod DV connection and this will be equals to the await we're going to use the connect function from the mongus so let's have the connect and we are going to pass our connection string so our connection string is process. EnV dot we need to have a connection string so let's go to the browser and let's have the mongodb

00:19:56 - 00:20:49

atlas so here you will see that I already logged in the mongodb at last where you can create different projects I have already created a project as react node 2 FAA okay so you can just click on the create project and then you can build your project here since I already have a project I will just go to the project and here you will see that uh if I want to connect to the project I have a connect new so I'm going to click on the connect new and here you will see that you have this connection strings so

00:20:22 - 00:21:16

what you can do is uh you can just copy so I will copy it I will go to my project and here I'm going to go and have connection string and this will be equals to the string that we have just copied so this is how it looks like now in this you just have to replace your password so this is a user you create whenever you create the project in the mongod DV at last so there you also need to provide a username and a password and you can just use your own username and a password to have this connection string

00:20:49 - 00:21:49

I will also create one more environment variable which will be the JWT secret because we are also going to use the uh Json web token so I will have my JWT secret as uh my JWT secret so these are the EnV variables that we are going to use it for our project so we have the connection string so I will use the connection underscore string here and this is going to help me to connect to my database now once I have this mongodb connection here I can simply do a console lock so that I can lock something and I know that my connection

00:21:19 - 00:22:19

is established so here what we can do is we can have database connected and then we can also use some value here so I can use here uh mongodb connection do connection. host and it is going to tell me that what host it is connected all right so now I am done with the DB connect function and what we can do is we can go to the index and we can just import it so what I will do is I will go here I will have the DB connect and since we are getting this app is crashed now we are using the es module so what I

00:21:49 - 00:22:43

will have to do is I have to use JS or I have to create my files. MJS now after doing the JS you will see that the application is running again okay so let's call the DB connect function so that we will see that our database is getting connected so I will go here I will have DB connect all right and I will save it and you see that as soon as I save it you see the database is connected and this is our cluster name

and this is our host all right so the database part is done uh with this database part uh I would also like to

00:22:16 - 00:23:20

create the model that we are going to use it for this project so let's go to the model and in the model what I will do is I'm going to create a model as a user.js all right and in the user do JS let's have import mongus and with this import mongus we are going to create the user schema so let's have the user schema and this will be equals to the new mongus do schema so we are going to create the user schema that we want all right we are going to have this object and the first part we need is we need to

00:22:47 - 00:23:45

have a username so this username will be an object it going to have a type as string uh we are going to use it as a required property because we need it and then we will Al keep it as a unique property so we are going to have the unique as true okay now with this we will also need a password so I'm going to copy this paste it here I'm going to change this to password type will be string required will be true and I don't need this as a unique property okay along with this uh I will also need a

00:23:17 - 00:24:24

flag which is going to allow me to understand that whether this user has MFA activated or not activated so I will have e MFA active and this is going to have a t type as Boolean and it is going to have a require as true oops required as false initially whenever we register a user the MFA will be not active so it will be a false and along with that we will also need to have one more property which will be two Factor secrate so whenever we are going to create a two Factor authentication for a particular

00:23:50 - 00:24:45

user we going to generate a secrate using the Speak Easy so we also have to store that secret with the user object so for now you will will just see that we are going to have this property and we are keeping this property as a string and this property is not required because this property will get value only whenever the MFA is activated so we are going to keep it as a not required property all right and along with this we are also going to use the time stamp so let's have the time stamp as true so

00:24:18 - 00:25:21

it is going to put the time stamps all right now once we have this uh what we will do is I'm going to use the constant user is equals to mongus do model and then we are going to pass the user here and then we have to pass our user schema so let's have the user schema uh I will go here and we will just make it as camel case so we have the user schema I'm going to copy I'm going to put the user schema here now we can just use the export default we can export this user so we have export default user so this

00:24:49 - 00:25:41

is how our model will look like we are going to have the username we're going to have the password we will have the ease MFA active and then we have the two Factor secret and then we have the time stamps so we are done with our model so this part is done I will just close it we also done with the database connection I will close it here we are almost done with all our middle Wares and configuration we will also need to provide the routes so now let's work on the routes part so I'm going to go here

00:25:15 - 00:26:28

I will create a file which will be Au routes.js all right I will import the router from the express then import the passport uh now we have imported the router and the passport let's create the router so I will have the constant router is equals to the router okay and then we can create our routes so I will have the registration route I will have the login route then I will have the a status route which is going to provide us the status of our user whether the user is authenticated or not then I will

00:25:51 - 00:26:51

have our log out route all right so this was the first part where we are going to have the four routes for the uh Au purpose so for that let's have the routes here I'm going to use the router. poost and this will be/ register and I'm going to have a register function here so I have to create the register function uh which will be created in the controller Parts where all the logic of the registration will be there all right I will copy I will put it here and I will change this to login and I will

00:26:21 - 00:27:27

also change this to login okay now for the third which is the OD status I'm going to copy here and I'm going to copy here and the O status would be a get API so we will just get the O status so I will going to have here as status and then I will have the Au status here and the last part which is the log out so

I'm going to have the log out and here I will have the log out function all right so we have all these four routes now we are also going to create the MFA routes as well uh now so I will copy and then I

00:26:54 - 00:27:58

will have it here so I will copy more okay so the first MFA route we wanted is the uh two Factor authentication setup so this will be 2 FAA setup which will be a post route and here the path will be 2 fa a/ setup and here we are going to create the function uh our function will be setup to fa all right the next route will be the verify token so whenever you have set up the 2 FAA you are going to get a token the time based OTP which you want to verify so I'm going to have here verify and this will

00:27:25 - 00:28:33

be 2 f/ verify and and the function will be verify to fa then the third one is to reset our two Factor authentication so I'm going to have the reset this will be also be a post this will be 2fa SL reset and then here I'm going to have the reset to so we have all these routes available here and all the functions uh we need to create so now let's export the router so I will have the export default and I'm going to have the router here all right so we have this now we can just import the router into our

00:27:59 - 00:29:07

index.js so I will go to the index.js and here I will have the app. use and here we will have the O oops API SL and then we can just provide the Au routes all right we need to import this Au rout so let's go and import it so I'm going to go here and here I can import so I will have import all routes and this will be coming from the routes slth routes.js okay so I have imported it but I still see that that I have an error and it says that register is not defined on the routes uh line number seven so if

00:28:34 - 00:29:32

I go here if I go to line number seven and yes uh we haven't defined the register yet so uh we can go to the controller and just Define all the functions so that we don't get these errors so I will go to the controller part and in the controller let's create the file first so I will have the Au controller.js okay now here we can create all the functions so first let's have export constant register which will be equals to a arrow function all right and I will have the async here okay now

00:29:03 - 00:30:10

I will just copy and I will put it for all the others as well so next will be the login then we are going to have the status or maybe it's the Au status so I will have the Au status then we are going to have the log out after log out we are going to have the setup to fa then we are going to have the verify to fa and the last one is the reset to fa so I'm going to have the set 2 FAA all right so now we have all the functions available here we can just import them in our route so I will have the import

00:29:36 - 00:30:43

and I'm going to import the register uh login log out out status setup to FAA verify to FAA and reset to fa and now we should not be able to get the error so if I go and check then we still get an error uh this will be because we have to make it at JS okay so now the application is back and it's working fine so we have written all the routes and we also have the corresponding functions into the all controller for our routes and in the register what we will do is we are going to get the request and the response so I will have

00:30:10 - 00:31:13

the request response and I will just copy them in all other functions as well so I will copy put it here here here so first let's have the TR catch Block in order to have register a user if it is an error we are going to have a response status as 500 and then we are going to give a Json uh object and we are going to have the error so let's have the error here and we going to have the error as error registering user and then we are going to give the message and here the message will be the error that

00:30:41 - 00:31:36

we have got so let's have the error here all right so this is what our catch block looks like in case of an error we are going to have the status code AS 500 and adjacent data with an error message and then whatever the actual message we get okay in the tryb what we are going to do is uh first let's get the D structure uh from the request body we get the username and the password so I will have a constant I get the username I get the password all right this will be coming from the request body so let's

00:31:09 - 00:32:07

have it from the request body now once I have that I want to Hash my password so let's have the hashed password and this will be equals to the await and we're going to use the bcrypt library so let's import that so I'm going to go here import bcrypt from bcrypt JS okay we can use the bcrypt now and here on the bcrypt we have the hash function so let's have the hash function and we pass

the password so this is the raw password and we need to pass the salt value in order to Hash this password okay we have

00:31:37 - 00:32:45

the hash password now let's create a new user so we have a new user and this new user will be of type user so we can just import the user so I will go here I will have a import user from Models SL user.js okay we have the user and this new user will be having a username yes we have the username this will be having a password and this password will be the hashed password this user will also have the EAS MFA so I'm going to have the E MFA active and the value will be false all right so we have created the new

00:32:11 - 00:33:09

user and you remember that there is one more field which is the two Factor secret which we did not provide it as required so that's why we are not providing it here but all these three fields are required now once you create this you can actually log this user as well so I will have the log user new user and we can just Define here as a new user so we can log the user okay now we can just save this user into the database so I will use the await and new user do save function so we have a save

00:32:40 - 00:33:50

function on it and this is going to save the user now if we want to just try this we can go and try I can go to the Thunder client here create a new request I can just remove this I will have the HTTP colon sl/ Local Host and I can just use uh here as register okay this will be a post API we can go to the body and in the body we can Define the user name I will just minimize this the username could be uh I will have the username as the page and then I will also have the password and I will provide the password

00:33:15 - 00:34:05

as 1 2 3 4 5 6 now I can send a request and we can just check so if I send a request what I see is this was the user and this is the new object and everything but we don't see any response so I will cancel the request so I'm not able to cancel the request what I will do is I will just kill my server all right and I will rerun the server okay and the mistake we did is we actually created a new user we also did it await but we don't know what is the next statement here we need to provide the

00:33:40 - 00:34:43

response back to the user so let's have a response status as 2011 and we also provide a Json response to the user uh that will be a message and we provide user registered successfully so we have this users registered successfully now we can give a try so before that we go to the database and see the user is created or not so I go back here uh I go to the browse collection and if I go to the users then we see that the user was actually created Created at updated at MFA false I will just increase the font

00:34:12 - 00:35:06

here so now what I will do is I will just delete it so that I can recreate the user so I will delete this and the user is deleted let's go and let's recreate it if I go here if I pass the same values and send the request then we see the user registered successfully and we can go and verify that if a user is created or not I I will just refresh it and we have the user is created so the registered user is working fine uh next we have to work on the login user where the user will be able to log in and then

00:34:39 - 00:35:36

based on the login we will also have this connect session ID and everything so that we can also see that so if I go here and if I go to the cookies then we don't have any cookies at this moment okay so let's go and let's work on the login part and for the login we are using the passport.js for the authentication purpose so we will also need to create the configuration which we just saw it on the passport.js documentation so let's go to the passport.js documentation first I will use the passport.org here we will use

00:35:07 - 00:36:02

the passport. local so I'm going to go here I will have the passport hyphen local we go here and we see that we have already installed the passport local and we have to build a configuration file where we can put this function and here how we can use it that okay let me Zoom this in the login route we can simply use passport. authenticate and we just provide our strategy since it's a local so we will provide the local all right and in case of a failure you can just also provide a redirection so let's do

00:35:35 - 00:36:37

that I will go back here I will go to the routes and in the routes we already have the passport so here I have to provide the passport middleware so I will have the passport. authenticate and here I have to provide my strategy so our was local all right so we have added this but just by providing this

passport. authenticate local our authentication will not work we have to create the configuration where we accept the username and the password and then we have to match the password with the password that is stored in our database

00:36:06 - 00:37:08

so that's how we authenticate and this is what we saw in the documentation as well we have to create this password. use where we will fetch the username and the password from the user then find the user and then we verify the password so let's do that so what I will do is I will go back here and in the config I'm going to create one more passwort config file so let's have a new file I will have the passport config.js all right and in the passport config.js I will have the import passport then I will

00:36:37 - 00:37:44

also import the strategy so I will have the strategy from the passport local given elas as local strategy now since we are going to receive a raw password from the user we have to match that with our hashed password which is in the database so I will also need a bcrypt library to compare those passwords so I will have the import bcrypt from bcrypt JS and in order to find the user into the database I will also need my model so let's have the import user from model so I will have the user.js now first we

00:37:11 - 00:38:06

have to use the passport. use which we can just either copy it from here or we can just write it our own so I will just copy it from here and put it here and we will just make the changes here so passport. use we are going to use a new instance of our local strategy which is going to accept the username and the password password now let's say you are using this passport local strategy but you are going to validate the user based on an email address and the password so in that case what you have to do is you

00:37:38 - 00:38:37

have to provide an object here so in this object you will provide the user name fill and you provide email address all right and then this can be changed to the email all right so this is how you will do it but since in our case we are just using the username so username is a by default property we don't have to provide this additional object here okay now once I have this username password then what I have to do is uh I'm going to use my TR catch block so let's have the try catch block all right

00:38:08 - 00:39:07

and in the try first we have to find the user so what I will do is uh I'm going to go here I will have a constant user and we have to find the user from our database so we are going to use the user model do find one and in this find one we are going to pass the username as username is a unique property for us okay since we are using using an AIT we have to provide an async so let's have a async here so I'm going to change this to async and this is an arrow function so here we have found the user from our

00:38:37 - 00:39:32

database once we have the user what we will do is if we don't find a user so let's say if there is no user then in that case I will just return and here you will see that done uh it's a very similar whenever we create our custom Middle where we have the request we have the response and we have the next so that once you are done with this you have to do that as a next part so that's how a very similar done is here so what I will do is once we have this I will do done and this done is taking first the

00:39:05 - 00:40:04

error so if there is an error you pass the error so in this case we don't have any error uh if you have a user object then you can just pass the user object but right now we did not find the user so this will be a false and then you can just provide your message so I will have the message and here I will have user not found as a response all right now if we found the user then in that case we have to match the the password so if we have is match this will be equals to the a weight we're going to use the bcrypt

00:39:34 - 00:40:40

do compare to compare the passwords we're going to take the raw password and this will be the user. password the password which we have got it from the database all right now once this is match so if is match is true then in that case what we will do is we are going to return uh the done function so in this the error will be null and the user will be the user object which we have got because the password is matched then we will just forward the user object but if this is not true then we are going to have the else and in this

00:40:07 - 00:41:06

we are going to return the done where we are going to have the null as an error we don't have a user object but this time we need to pass a message so let's have a message and the message will be incorrect password and if there is an error then in that case what we will do is we are going to return our done function with an error okay so this is done uh I will just remove all this now we don't want all this stuff uh as we have already return our function all right so this is how it looks like we

00:40:36 - 00:41:28

have this passport uh use new local strategy instance where we have the username password and we see that oh this is wrong this should be password so that's why we have an error here okay we have the password and then we have the user password and this is done now if you want to give a try for this uh we can just do that what we can do is we go to the Au routes we have passport. authenticate and we have the local here and we have this login so we will just have to write this login function in the

00:41:03 - 00:42:04

Au controller so let's go to the Au controller and here what we will do is uh I'm going to have the console. log and we can just have the authenticated user is and we can have the request user so here we are going to get the request user object because what we are doing is in the passport once this is done and everything is authenticated we are actually forwarding the user so when we forward the user we get get that user values here and here what I can do is I can have the response status as 200

00:41:34 - 00:42:44

because it was a successful authentication and then here I can Define my message and here will be user logged in successful so I will have the user logged in successfully and along with that I will also Define uh the username which is has logged in so I will have the username here which will be the request user username and we will also going to pass the ease MFA active or not so I will have the request user do is MFA active and the reason we are passing on the login is mfi active is based on this we will be able to know

00:42:09 - 00:43:07

that the first part of the authentication is the done and the second part of the authentication is required so if the MFA is false then from the front end side we are going to call the setup two Factor authentication and if it is true that means the two Factor authentication is already set up we just need to verify the token so we're going to see it later on when we actually test all our apis for now

we will just test the login part so let's go back here I'm going to change this to login and for the login also uh we are

00:42:38 - 00:43:46

going to pass the same thing uh the username and the password and let's give a try so what I will do is I'm going to send the request and we see an internal error and that is obvious because we did not included the configurations of the uh passport config into our index.js so let's go here I will have import and we need to have do do slash or maybe do slash we go to the config then we go to the passport config.js all right now if I go and check the everything is working fine let's go here and I will just send

00:43:12 - 00:44:07

a request and we see an internal server error so let's go and let's see what is wrong so here you will see that uh we have if we go here we see that we have this failed to serialize user into the session all right so when we are using the passport config so this passport config uh we have the particular user which we have found and we have authenticated it but now in order to put this user and create the session for this user we have to serialize this user so for that what we can do is there are

00:43:39 - 00:44:45

two function serialize and deserialize so that's where passport and the session they both work hand inand usually and we can provide here the passport. serialize user okay and in the seriz user uh what I will do is uh we are going to have the user and we have the done function all right and then what we will do is we just have to use done and we have to pass the null as an error all right and then the second one is the ID that we have to pass since we already have the user we also have the user ID which is

00:44:12 - 00:45:14

the user ID so this user ID if we go and check in our database then we see that we have this underscore ID of the object so that's where we use this underscore ID now we can also go and do a console log here console. log and what we will do is we going to have a we are inside serialize user all right so that we know that at what time we go into this function now with this serialized user we also have a deserialize of the user so what I will do is I'm going to copy put it here I'm going to change this to

00:44:43 - 00:45:47

deserialize user and in this deserialized user we have to pass the ID that we have serialized so this is the ID we have passed so I'm going to copy and put the ID here okay once we have this what I will do is I will have the pry catch block and in this we are inside the serialized user all right and here we will just find the user with an ID so I will have a constant user I will have a wait user. find by ID and we pass the ID in order to identify our user all right once we get the user we are going

00:45:15 - 00:46:14

to call our done function where error will be null and we pass the user object and if there is an error we will again have a done function and we just pass the error so we have done the serialize user and deserialize user now let's go and let's give a try so what I will do is I will go back here and this time I will send the request again and we see that the status is 200 and here we can also see that uh whenever we call it actually we are inside a serialized user so the user is serialized here okay now

00:45:44 - 00:46:46

if we go and check our response if we go and go to the response here then you see that the user logged in successfully username was divish and MFA is not active it is a false if I go to the cookies you will be able to see that since we have serialized the user we have established a connect session ID and this is our session ID and the data we have so that's how we have established the connection and the session now since the user is logged in and we also get the values of the response as the user name and the MFA

00:46:15 - 00:47:16

active we can go and work on our Au status API and then we can also work on our log out API so let's go to our Au routes now since we have the session established is so for every subsequent request that we are going to do from our Thunder client we are going to forward the cookies as well and the reason behind is that in the index if you remember we have used credential as true so whenever a request from any of the origin comes we will also get those cookies and once we get those cookies we

00:46:46 - 00:47:48

actually get the user information from there from the cookies okay so let's go to our controller or maybe just go to the route and we will go and work on the OD status so in the OD status we have this

function so let's go to Au controller and in the Au controller we have the request and we have the response so here it will be very simple for us that if we have the request. user if that is available then in that case we are simply going to have the response status as 200 and then we are going to

00:47:16 - 00:48:17

pass the Json object as a response and we are simply going to pass the same thing so that will show us that okay the user is authenticated all right but if this is not then in that case we are going to have an else part and then we are going to have a response status as 401 and we are going to have the Json so let's have a Json and then in the Json we are going to have the message and we say unauthorized user okay so we have the unauthorized user and this is what the status API is going to give us we

00:47:48 - 00:48:41

can also write the log out at the same time so for the log out what we will do is uh we come here and then we will here see that if request. users or maybe I will just simply copy this so when we are trying to log out and in that part if we see that the request is not having the users that mean the user is already not authorized to log out he's already logged out so we have this and if this is not the case then in that case what we will do is we are going to have a request so there is a log out method

00:48:14 - 00:49:17

here so we are going to use this log out and then what we will do is if it's a call back function uh we are going to have that if it's an error then in that case we are going to return I'm going to copy this and I'm going to put it here we are going to return 400 and we say that user not logged in so that means the user is not logged in so I will have the user not logged in okay but if this is not the case then in that case we are simply going to have a 200 okay the log out is successful and we're going to

00:48:45 - 00:49:47

have a Json and here I'm going to have log out successful all right so now we can give a try uh for these two apis uh let's go here and now since we have already logged in uh if we want to check the status we can give a try so let's have a status and I'm going to go here get send then you see that unauthorized user since we have already restart our application because we are working on it let's give a try to login again first so I will have the login post send and I see it's a 404 because it's log in so I

00:49:17 - 00:50:16

missed the L I'm going to send the request and we see that the user logged in now let's go and check the status if we want to check the status I will go here make a get and I will send then I see that this is the status of my logged in user if I want to log out I can just have a log out I can create a post API and I don't want this information and I click on send then you see the log out successful and whenever you log out successful you will see that we are into deserialize user so that's where when

00:49:46 - 00:50:37

you log in you actually have to serialize the user into the session and when you log out you actually have to deserialize the same user from the session so that's how the login and log out is work let's go and let's get the status again since we have logged out we should be getting because there is no logged in user we should get unauthorized or something so we see that we have unauthorized user so this was the first part of our application where we have did all the setup for the passport we

00:50:11 - 00:51:13

did the database connectivity we did the registration of the user we Ed the login using the passport local strategy we have the Au status of the user we also have a log out feature this was the first Factor authentication the second part is now start building the second Factor Authentication or we say multiactor authentication let's go to the Au routes and we will make start from the 2fa setup first so for this uh the first thing we have to do is uh these routes the 2fa setup verify and reset are only accessible to an

00:50:42 - 00:51:43

authenticated user because we have to check that the first Factor authentication is done if that is done then only you can access these routes so we can have a small middleware here so what I will do is I will have a request response oops uh request response and next and this can be an arrow function I will have a comma here so this is a small Middle where I'm just writing it in here and you can also create a separate file and include this middleware but what I'm going to do here is that if the request has an ease

00:51:12 - 00:52:18

authenticated so I will have the ease authenticated and if this is true that means the user is authenticated I will have return next that if it's an authenticated you can just go ahead and set up

your two Factor authentication but if it is not then that case I will have the response status as 401 and then Json object message and the message will be uh unauthorized all right so we have this unauthorized and what I can do is I can just copy this middleware in all my routes so I will copy here and I will

00:51:46 - 00:52:55

copy here okay so we have it so that means I mean if I try to go and access it now I will have the request is not authenticated because if I go and check the status I'm unauthorized so even if I go to uh to fa slash and if I try to do a setup and it's a post and if I do it I will get unauthorized because I'm not authorized if I want to give a try I can go to my Au controller and here I can just have a response. Json uh and I can have message hello world okay let's give a try I can go here let's login first so

00:52:20 - 00:53:20

I will go and login I need to provide my username page I will provide the password as one 1 2 3 4 5 6 and let's give a try I'm going to send the request I see user the p is logged in successfully now if I want to set up a 2of so I will have 2fa slash and here I'm going to have the setup and it's a post so I'm going to just remove this then I should be able to see the hello world you see I see a hello world but if it's not an authenticated so let's say we call a log out first so let's have a

00:52:50 - 00:53:51

log out and I'm log out and now if I try to call 2fa SL setup I will get an authorized okay so that's how we are just doing it with the help of the small middle we writing it here okay now since you are already authenticated you need to set up your 2fa so let's go to the controller and write the 2fa function so what I will do is I will start with the tri catch block here and in the TR catch block uh let's have the error case first so I'm going to just uh copy from this I will copy this I will put it here all

00:53:21 - 00:54:22

right and I will say that if it's an error so it's better I copy it from the uh register users so this is good I'm going to copy let's go into fa put it here yeah if it's an error then in that case what I will do is uh I'm going to have here error setting up tofa and then we have the error message now if it's not an error then in that case what we will do is first let's have a console. log and we will have the request. user is so here we will find all the details

00:53:51 - 00:54:55

of the user in the request user now once we have the request user details uh what I will do is I will have a user from my request user I take the user now after taking the user uh we have to create a secret for the multiactor so if we go uh let's go to npm speak easy and if we want to generate a secret first so the first part is to generate a secret we can simply generate a secret like speak easy. generate secret and this is going to return an object with a secret asky value the secret hack value secret base

00:54:22 - 00:55:29

32 and it also Returns the OTP o URL which we can just use it and then create a QR tag okay so we are going to follow this step first so I'm going to copy we go here and we generate a secret now if we want to log in and check what are the secret object contains uh we can just have the console.log the secret object is and then we can just log this object so let's have the secret okay so we have the secret object now once we have the secret object what I will do is we are going to have the user Dot and if you

00:54:56 - 00:55:59

remember we had this uh property on the user object which was this two Factor secret so now we are going to assign that value here since we have generated a secret this secret has this value of of secret do base 32 all right so we have this and we have generated this now if you want to give a try we can just give a try this uh what is the value of this object is okay so what I will do is I will go here uh I will first have to log in okay to log in I will have to give my username the page and I will

00:55:28 - 00:56:41

give the password so let's have the password 1 2 3 4 5 6 okay I will send the request login is successful I will just copy and create one more request this will be 2 f/ setup and it's a post all right I will send the request then we get the error setting up to fa okay that's fine uh we go and see the request user is undefined and if I go and check the status so I will just copy and I will have the stat then we see the user is logged in so it should work so I will have a 2fa SL setup and it's a post okay

00:56:04 - 00:57:02

so we get this error if we go and then I can see that we have this rest it should be a request all right and now we can give a try so first let's me login so login successful and now if I go and send then we still get an error that's fine but here we see that now we have the request. user and we get that user

if we go and see that then we have this speak easy. generate secret which is not getting generated and the reason is we did not imported it so let's go and import that first so I will have import

00:56:32 - 00:57:35

Speak Easy from Speak Easy okay I will also need to import the QR code so let's have the QR code from QR code and then I will also need to have my Json web token so I will have the import JWT from Json web token all right so I have imported everything what I needed uh now we can give a try so I will first log in logged in successful I will send the request and it will not come anything but we can go and see here what we get so here you will see that if I zoom then you see that this is the secret object we have

00:57:04 - 00:58:08

the ask key we have the hex base 32 string and then we have the OTP o URL so this is the OTP o URL you can eventually store all these the hex code and everything but I'm just going to store the base 32 into my two Factor secret so let's cancel this I will go here and I will minimize this let's go here and I'm going to put this base 32 Secret into my two Factor authentication all right so once I have this I will need the same two Factor secret whenever I verify my OTP as well so after this what I will do

00:57:35 - 00:58:35

is uh I will have my user. e MFA active and now I'm going to set it as true because the MFA is now active and I will have a wait user Dove so this is going to save the user into the database with the latest value and those latest value will be two Factor secret and is MFA as true all right so I have saved it and after saving what I have to do is I have to generate a URL which is a custom URL so here if you see that then there is already a URL but I have to make this as a customized URL so what I can do is I

00:58:06 - 00:59:16

will have the URL here this is equals to speak easy. OTP URL and in this I can just pass some customized values let's say I want to pass the username when I scan the QR tag I know that okay this is the user so I will have an object here and in this object I can just pass all the values so first let's pass the secret I have the secret. base32 then I can also pass the label so for the label I will just pass my username so whoever the user is logged in and is setting up the MFA so I will have the request user username okay

00:58:41 - 00:59:54

then I can also pass the issuer who is issuing this so I have the issuer and the issuer will be maybe pia.com okay so I can just pass the issuer name here and at last uh we need to have the encoding so I will have the en coding as base 32 base 32 okay so we have this URL now we have to pass this URL into the QR code uh so that it can generate an image or a base 60 post string so I have this QR code Library dot and this is having two data URL so you have just pass the URL here so I'm going to pass the URL and let's have a

00:59:17 - 01:00:28

constant here so I have a constant QR image URL is equals to A wait okay now once this is generated I can just pass the response so I will have the response status as 200 MFA is set up do Json and in the Json what I will pass is I will pass the secret if you don't want to pass secret you don't pass it but I can just passing a secret in case you want it but it's not recommended so I will have the secret. base32 and the other value which I want is I will pass the QR code and I can have the QR image URL all

00:59:53 - 01:00:58

right so now if you want to give a try after setting this up let's give a try so what I will do is uh I'm going to go here I will log in so user is logged in uh let's set up the MFA I'm going to send the request and you see that we have this secret now along with the secret we also have this QR code base 64 string now I will just copy this base 64 string and then we will just use it so I will just copy the base 64 string all right now if I go here and I type uh base 64 string to an image let's try

01:00:26 - 01:01:19

this one I will just put the and you see that as soon as I put it it generates a QR tag and this QR tag actually contains the values so if I want to scan this what we can do is I'm using this extension which is the authenticator so I'm going to click here and you see that this is the authenticator I will just remove all the old one so I'm going to remove okay now I have nothing okay and this is uh basically a Chrome extension for authenticator so maybe have this one so this one you can just simply download

01:00:53 - 01:01:47

it and put it into your Chrome extension yeah for testing purpose so now I have this one what I will do is I will click here I will click on and I will just scan my QR code so I will scan this QR code and as

soon as I scan it you see that the page has been added and if I click okay and if we go here then you will see that this is the issuer ww. the.com and then you have the username and then you get this OTP now you need to take this OTP and then you verify so that's how you do the second Factor

01:01:20 - 01:02:13

authentication so right now we don't have a verifi API so let's write that API first so that we can verify so I will we'll go back to visual studio code let's go to the controller and let's work on the verify all right so now in the verify what we will do is first let's have the token and we get this token from the request body once we have the token what we will do is we are also going to get the user which is in the request user because all these end points are authenticated so we will

01:01:46 - 01:02:49

always have the user from the session now if we want to verify what I will do is uh I will have the verified and we will use the same Speak Easy do T OTP function which is the time based OTP function uh and this has do verify so we want to verify it and we have to pass few values here so first value will be the secret and this secret will be the secret which we have it in our database that is the two Factor secret okay that's why we store it in the database next one will be the encoding we use the

01:02:17 - 01:03:32

encoding as uh 32 so we pass the same 32 and then we pass the token all right now if this is verified so what I will do is if this is true and if it's verified then I will pass my jwe token to the user so that user can use it for further access of the apis in the project okay so I will have the JWT token is equals to JWT do sign and we can pass the username as user do username okay and then we will also need to provide the JWT secret which we have added in our EnV so let's get that so I will have the

01:02:55 - 01:04:21

process s.v. JWT secret okay so we have this and now we can also Define uh the expiry time of our token so I will go here I will have the expires in and we will use 1 hour so we got the JWT token now once we got the JWT token we can just give this token to the user as a response so I will have the response status 200. Json we will have the message and here I will have 2 FAA successful and then give the token and the token value will be the JWT token okay but if this is not the case then we have to give the response uh in

01:03:37 - 01:04:41

the else so if it's not okay then we have the response status as 400 because user is giving us a wrong token so that's why we will have an invalid uh 2fa token so let's have a message and in that case I will have invalid 2 FAA token all right so we are going to test this uh but before testing this what I will do is I will just quickly write the reset function as well as we already know if you want to do a reset the first part we do is uh we change this MFA active as false and we just remove the

01:04:09 - 01:05:16

value in the two Factor secret okay so we go here and here what I will do is I will simply have the try catch block uh in the error I will have the response status as 500 dot Json and we are going to define the Json uh error this will be error resetting toofa and second one will be message and we are going to define the message as error but if this is not the case then we have to reset it so what I will do is I'm going to take the user first this user will be coming from request user all right and on this

01:04:43 - 01:05:47

user we will first do the user two Factor secret we will make it as an empty and on this we will have the user is MFA and we make it as false all right once we do this we have to save the users so I will have the user do save and then we can just give the response back so I will have the response status 200 and then we're going to have the Json message and let's have the message as 2 FAA reset successfully so I will have the successful copy and put it here so we have the reset to FAA successful

01:05:15 - 01:06:15

we have also returned the reset and the verify let's go and let's check for the verify so what I will do is I will go here I will have to log in the user first so log in once the user is login the MFA is true so we can just reset it so what I will do is I'm going to go here and I will click on reset and send and you can see that 2 FAA reset successful if I go again here and if I want to have a status so let's have a status and let's change this to get I will send it then you see that the user

01:05:45 - 01:06:44

is logged in successfully we are already logged in and our e MFA active is false so that means we can set up again so I'm going to click here I will send and we get this basic 64 string so let's copy the base 64 string I'm going to copy go to the browser and first I will just remove my old one so let's remove the old one okay uh I'm going to clear this and we put our new base 64 string and we get this new QR code now we can go here and I will just scan this QR code so I'm

01:06:14 - 01:07:11

going to scan the QR code here okay the page has been added so I'm going to click okay and we see that we have this OTP now I can just copy this OTP and if I go here and instead of the setup I'm going to use verify okay I will go to the body in the body I'm going to use token and I'm going to pass my token value I will send the request and you see that invalid MFA token if I go back and I take the new token okay I'm going to send then you see that 2fa is successful and you will see that we have

01:06:43 - 01:07:36

this jwd token which we can just use it if we go and check the status it's MFA false but since we have set up now we should get it as true so this is how you'll see that we have established the user session we have validated the user with the username and a password then we have also checked the status of the user and then we have set up the 2fa so this is a thunder client I'm using we can just go and try it on the postman so I will open a postman and I will create a new user on that uh this is a new user

01:07:10 - 01:08:08

uh let's create it so what I will do is uh I'm going to copy this I will put it here and I'm going to first register a user so let's have register I'm going to change this to post uh I will go to the body and then we can just have a username so let's have a username as uh Nikes okay okay I'm going to send the request and we see the user registered successfully if we go and verify in the mongod DV then we should be able to see so if I go to the browser I refresh it then now we see that we have two users

01:07:39 - 01:08:37

we have Nik and the MFA is false let's activate that so I will go back here I will go here and I will first have to log in the user so log in send and we see the user is logged in is MF active is false so now if I also want to see the cookies so I can go to the cookies and here you will see that we have this one cookie which is the connect session ID and everything so we have established a session here now what we will do is uh I will go here and set up so I'm going to click on send and you'll see that we

01:08:08 - 01:09:12

have a new secret and this is a new uh if I just copy let's go to the browser I will go to the browser I will just remove everything I will put it here and we see that we have an error so I will have to

remove the other part of the response so let's remove that now after removing I get a new QR code now let's scan this so I will go here I will click here and now I will just scan the QR code and you see the Nik has been added so we see that we have here two users one is the P one is Nik and let's say if

01:08:39 - 01:09:38

I try to validate from the p uh OTP into the Nik so what happens so it should not work so what I will do is I will just copy I will go here I will use verify go to the body I will remove everything I will use token I will use the token toen value and post and I will send then you see that invalid MFA and that is true this is what is expected because we are using a token from some other user let's go and let's get the token from Nik so I will just wait until I get a new token okay now I get a new token I'm going to

01:09:09 - 01:10:06

copy and I'm going to use it here all right now I'm going to send then you see that this is now successful uh similarly if I want to do a verify for the page as well I can just take a token for the page and put it here and I can send and I see that I have two users who have logged in in two different clients and they both have different session established so that's how you can actually create this multiactor authentication and which is so cool project because there is a lot of things

01:09:38 - 01:10:26

we can learn here so maybe we will take a pause here we have completely built the backend part of our multiactor authentication we have all our apis ready we have also tested our apis from different clients like a thunder client or a postman so I hope you enjoyed the project if you like the video a thumbs up is appreciated you can also connect with me via Facebook or Instagram you can follow me on Twitter for latest updates and before you go don't forget to subscribe the channel and press the

01:10:02 - 01:10:12

Bell icon so that you don't miss the videos like this one thank you thanks for watching