Create project folder memories-project

Create 2 subfolders client, server

In client folder create react application by

*Npx create-react-app ./*

It requires npx, node etc. hence for windows we need nvm for windows, download it using the installer provided on their github page.

Then install additional necessary files using

*Nvm install latest*

Then you will need nvm to use a version of node by running

*Nvm use latest*

But to run this command you need to run command window in administrator mode

At that point you can finally run

Npx create-react-app ./

Then we move on to server folder, wecreate index.js, it is going to be the starting point for our backend application. In server directory run

*Npm init –y*

It is going to initilaize empty package.json hence we can move on the install our dependancies, they are

*Npm install body-parser cors express mongoose nodemon*

Body-parser : enables us to send …. Requests

Cors: enable cross origin requasts

Expres: fr\amework for creating the routing of our application

Mongoose: we're gonna use mongoose to create models for our posts

Nodemon: we're gonna use nodemon so that we don't have to manually reset the server every time we make a change

In index.js imports these modules, makes some adjustments to package.json,

Now we go back to cliemt directorty and install necessary dependancies for it as well

*Npm install axios moment react-file-base64 redux redux-thunk*

Axios: we're going to use axios for making api requests

Moment: is a library for working with time and date

React-file-base64: we'll use this to convert our images

Redux-thunk: used for asynchronous actions using redux

He mentions that in the client folder there is src folder with some files, but he says they are not all necessary hence he just deeltes it and creates a new empty src folder and creates index.js file in it. Index.js file in client is gonna connect the react application to index.html file. Writes to index.js

Then creates App.js in src

Writes to it

then start the applicartion with

*npm start*

first we are going to set up our back end. we need to connect to the database create a mongodb cluster set up the models and also create some documents

writes to index,js

creates mongodb atlas cluster

initialllty writes mongodb connection string right in the codew but says that

of course in real applications your credentials should be secured so we cannot simply p ut them in here later on in our video before the deployment of the whole application

we're going to create environmental variables and then we're going to store that connection url right there the second thing we need is going to be port so const port is equal to process that env that port for now we're going to use the 5000 but later on once we push this to heroku heroku is automatically going to populate environmental variable called port

mongoose,coonecrt() returns a promise

don’t forget that to cobnnect to mongodb atlas you may need to whitelist your development pc’s ip.

A t this point I could connect to database.

Then we go on with routing, we create a folder called routing.

In that folder we create posts.js file which specifies the routes for our application, it details the response funciotns for http requests to specific routes. The response functions themselves can be defined in a separate file in a separate folder to reduce too much wording in the same file. Hence we also create under server a directory named controllers, and in that we create posts.js as well. After populating thiese files ew also need to import it in index.js and use that router defined in posts.js.

Then we go on with mongoose models. Create models folder, and in that create gfile podstMessage.js.

Populate it,

Then in posts.js adds functionality for get posts and create post, then moves to the client sifde.

npm install @material-ui/core –force

we form the directory structure of clientside.

We then create the page style utlilizing material ui module and coding in App.js

Insid e App.js we have used some react compnents named post posts and form, hence wwe go ahed to define them, we create components folder and create a convenient folder structure, and populate these files. Then we make a test run to see if everything is working.

Then we add styles files for App.js, and other component files. We use prepared styles to not lose time on them.

Then we go to api side of client. We gonna use redux. We create ations and reducers folders. We popoulate an initial version of redux related files, we also install react-redux

Then qwe move to the form component.

UseState from react ???

UseDispatch from react-redux ???

Then we add the api post request for form

Then we can with ui submit a memory, meaning that we can correctly send http post, and it is actually stored in database.

Here video part 1 ends. I continue with video part2.

We go on with posts and post component

Then we can test and see that posts are displayed on web page.

Then we add background with index.css.

Then we go on to backend to add edit post functionality.

Axios???

Then we also edit client side for edit post functionality

Then we add delete functionality.

In backend we update routes/posts.js

Then implement the function itself in controllers/posts.js

Then we go to clientside api/index.js and add delete function

Then in actions/posts.js we add delete functionality

Then we go to reducers/posts.js and add delete function

Then add on click function to post component delete button

Then we add like functionalty just like delete functionality

Themn we examine how web page looks on mobile devices using browsers inspect function

Then we arrange enviremnt variables,

Install dotenv

Add port and mongodb atlas connection url inside .env

.env file will not be uploaded online, but to make developers know which environment variables are nececassary, he suggest to create .env.example file with required variable names and explanations

Then deployment comes.

he deploys backend on heroku

he deploys clientside on netlify

I may try to deploy them to AWS EC2

VIDEO part 2 ends here.