**MERN STACK WEB DEVELOPMENT**

**Mongo Db atlas :**

It's actually a document database which is a certain type of no SQL database and it's very similar to Jason syntax or are javascript objects. So it pairs really well with javascript or node applications. Now, you have the option to install Mongodb locally or in the cloud using Mongodb Atlas. Now we're using Atlas, we're using a cloud database for a few reasons. One, it's easier to manage. We don't have to go through an installation and dealing with, you know, local database issues. two, this is a course. So some of you using Windows, some Mac and Linux and it's different on all systems. three, we're going to be deploying to Heroku, which is a platform as a service. So it, they, you can't install a local instance of mongo, when you're using Heroku, you'd have to use something like DigitalOcean or something like that if you wanted to actually install mongo on the server. So we have a cloud database that we can just connect to our app no matter where it's hosted locally or on Heroku or anywhere else for that matter. Right. So, those are the reasons that we're using Atlas. However, if you want to install it locally and use it, you can, our, the code will be the exact same. You'll just have a local connection string instead of a remote on

Ok so after creating a project in mongo db atlas

Build a cluster : username : simalbutt15 and password: adDI2elGfIvmwjoc and ip address: 103.156.153.37

Connect to your mongo db deployement : mongodb+srv://simalbutt15:adDI2elGfIvmwjoc@devconnector.rdmoq6x.mongodb.net/

**MONGOOSE:**

it basically allows us to model our data and it gives us a bunch of methods to interact with our database. So we can call like dot find or dot um remove or whatever we want to do with our database and it makes things easier**.**

**Git:**

**To initialize git repository :** git init

**To keep track**: git add .

**For commit** : git commit -m "Normalize line endings"

**Express server :**

Web framework for backend.So we need to create the package.json yo do that write npm init

**Regular Dependencies:** npm install or I and of course

* express, which is our main web framework for the back end
* express validator for data validation. So when, when we make a post request to our API um if there's fields that need to be there that aren't, it'll raise an error. So we'll use that for validation
* bcrypt. We're gonna use BCRYPT JS which is used for password encryption. You never wanna store plain plain text passwords in your database.
* config for global variables.
* gravatar for profile avatars. How that works is if a user signs up, they can use an email that's associated with a gravatar account and it will automatically show their profile image.
* Jason web token because we're using JWT to pass along a token for validation. They are used to transfer private data in code eg btw client server

Header: Specifies the algorithm used and token type.

Payload: Contains the data (e.g., user ID).

Signature: Used to verify the token wasn't tampered with.

* Mongoose, which is a layer that sits on top of the database so we can interact with it
* request, which is just a small um module that will allow us to make HDP requests to another API. And the reason we're installing this is for github repositories, we want our profiles to be able to have github repositories listed on them. So we're gonna make that request from our back end so that we can hide our API key and stuff like that and we can just return the repositories.

**Dev dependency:**

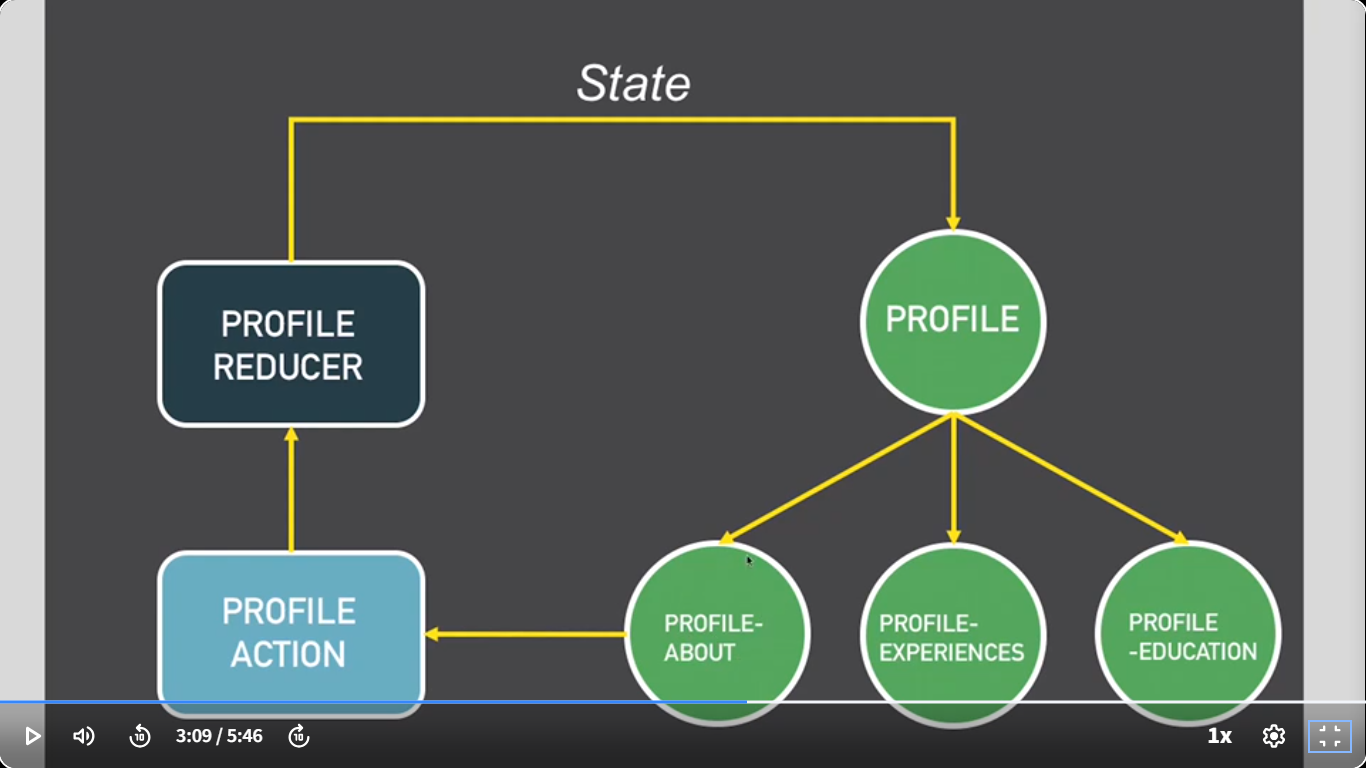
* install dash uppercase D
* Nodemon, which will constantly watch our server so that we don't have to refresh it every time we make a change.
* Concurrently which is going to allow us to run our back end express server and our frontend react DEV server at the same time with one single command

**React dependancies:**

* axios to make HTTP requests
* Redux, React dash, Redux,
* Redux thunk, which is Middleware to allow us to, to make a synchronous request in our actions.
* Redux DEV tools and there's a package we can use to make that a little easier. So Redux dash dev tools, dash extension.
* moment which is a, a date and time library to format date and time. And then we also want React dash moment. So we can use moment within the component.

**Redux:**

Redux is a state manager



[React Component]

│

▼

(dispatch an Action)

│

▼

[Action Object] ← Example: { type: "LOGIN\_SUCCESS", payload: userData }

│

▼

[Redux Middleware] ← thunk intercepts async actions (API calls)

│

▼

[Reducer(s)] ← rootReducer -> userReducer / postReducer / etc.

│

▼

[Redux Store State] ← (Managed by configureStore)

│

▼

[React Component re-renders with updated state via useSelector()]