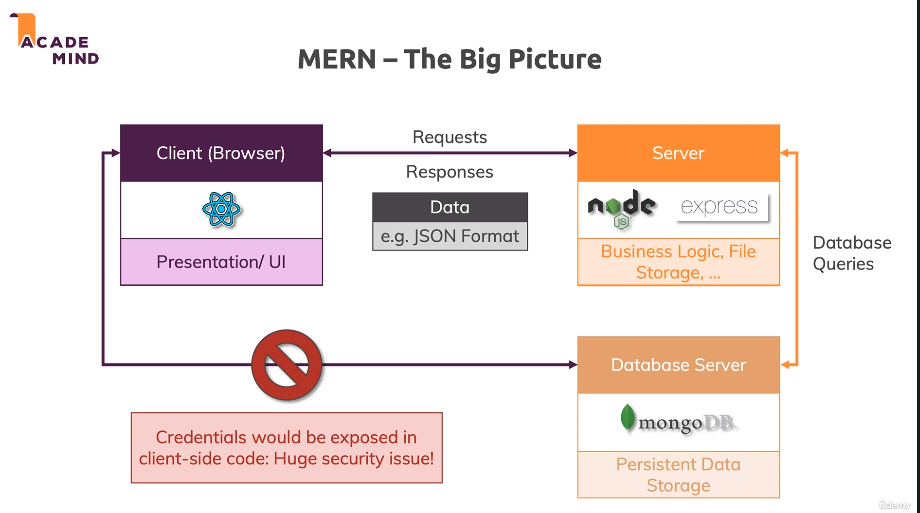
**MERN Architecture**

****

**MERN Stack Architecture**

This architecture allows us to build amazing web applications, highly scalable and very fast web applications with a beautiful and highly reactive user interface powered by React attached to them.

**React: (R) [FRONT END/ CLIENT SIDE]**

The React is responsible for the client side, for the browser side for what the user sees, so the presentation and the user interface.   
This means that we will use React.js to build our frontend facing web application, the thing that runs in the browser. It’s a Javascript library and its javaScript that is executed in the browser.

So React is used to build beautiful and highly reactive user interfaces in the browser, it’s there to render something onto the screen and re-render it whenever something changes, update the user interface and provide a greate user experience to our end users, put in other words, React is responsible for what the users sees in the end. React alone is great and we can build stunning user interfaces with it but of course if we only work with React we have some limitations, most importantly we’re not really able to execute any logic on a server, so in a place where users can’t see our code, they can see it in the browser, anyone can use the browser dev tools to look into our code. In addition to that we typically also want to store some data in a persistent storage and the browser side is not such a persistent storage. The user can clear the data there, the browser might clear data on its own if it’s running out of space, so data there is not persistent. It’s also not shareable across all users of your web applications because if store data in the browser, of course it’s only readable by that browser, so in the end by that user and not of other users of our web application.

**Node & Express: (N & E) [BACK END/ SERVER SIDE]**

This is a web application that runs on a dedicated machine, a server, somewhere in the Internet reachable by anyone, so opened up to incoming Internet connections and that server is created and run with the help of Node.js and a Node Framework i.e. Express.js.

These two pieces are used to write JS code that runs on a server, detached from our client, from our browser.

We also can use Node and Express for files storage.

Not this server side and the client side, so the browser, communicate with requests and responses, HttpRequests and responses, specifically so-called Ajax. Requests and response is triggered from client side JavaScript so that they are sent to the server and the response is handled in the client without reloading the page on the client, that is achieved by exchanging data which is not a HTML page which would be rendered by the browser and which would therefore lead to a page refresh but instead which is in a so-called JSON format. JSON is by far the most common format for exchanging data, JSON stands for Javascript Object Notation and its’s a machine readable and also quite human readable data format which in the end is used to exchange text data, numeric data and structured data in any form.

This data is attached to both requests send from the client to the server and responses received by the client, incoming from the server to then re-render something on the clinet side, so using React to re-render parts of the UI or to do something on the server side if the data’s received there, for example store it in a database.

**MongoDB: (M) [Database Server]**

We also have an extra database server which runs our MongoDB engine.

There are 3 big blocks working together – 2 servers, one running our Node-Express app and one running database engine and the client side.

Now that data server can run on the same machine as our Node Express server or on a totally different machine, that doesn’t really matter.

The database server in MongoDB is then used for the persistent data storage, not file storage, you should always store files on a files system, not in a database but any other data, like the name of a product, the price of a product, the users of our web application, things like that would be stored in database like MongoDB.

Who is talking to the database server?

That our Node Express Application. That application sends database queries using the MongoDB SDK or MongoDB library to be precise, that database server.

**Note:**

We don’t’ send requests form the client side directly to the database. Because to send these queries, we need to include our database credentials, so the user name and password to log into our database so to say.

As we know all the code that runs on the client side, so in the browser, is readable by all our users, there is no way of disabling this, how the web works. So if that code would include the credentials for your database, our users could hack our database, they could gain access to it.

So we do this on the server side where the code is not readable by our users and from the client side we just requests and responses to the server and on the server, we decide which requests who want to handle and which responses we want to send back, wo we have full control on the server side and the client is only able to communicate with the server within the requests response patterns we allow.