**REACT**

A Client-Side Library which allows you to build highly reactive user interfaces

* Render UI with Dynamic Data
* Handle User Input
* Communicate with backend services
* Provides a ‘mobile-app’ – like user experience

**NODE JS**

A server-side runtime: JavaScript on the server-side

* Listen to request and send responses
* Execute server-side logic
* Interact with Databases and Files
* An alternative to PHP, Ruby on Rails, Java etc. Is rarely used Standalone

**Express (Node js framework)**

A node framework which simplifies writing server-side code and logic

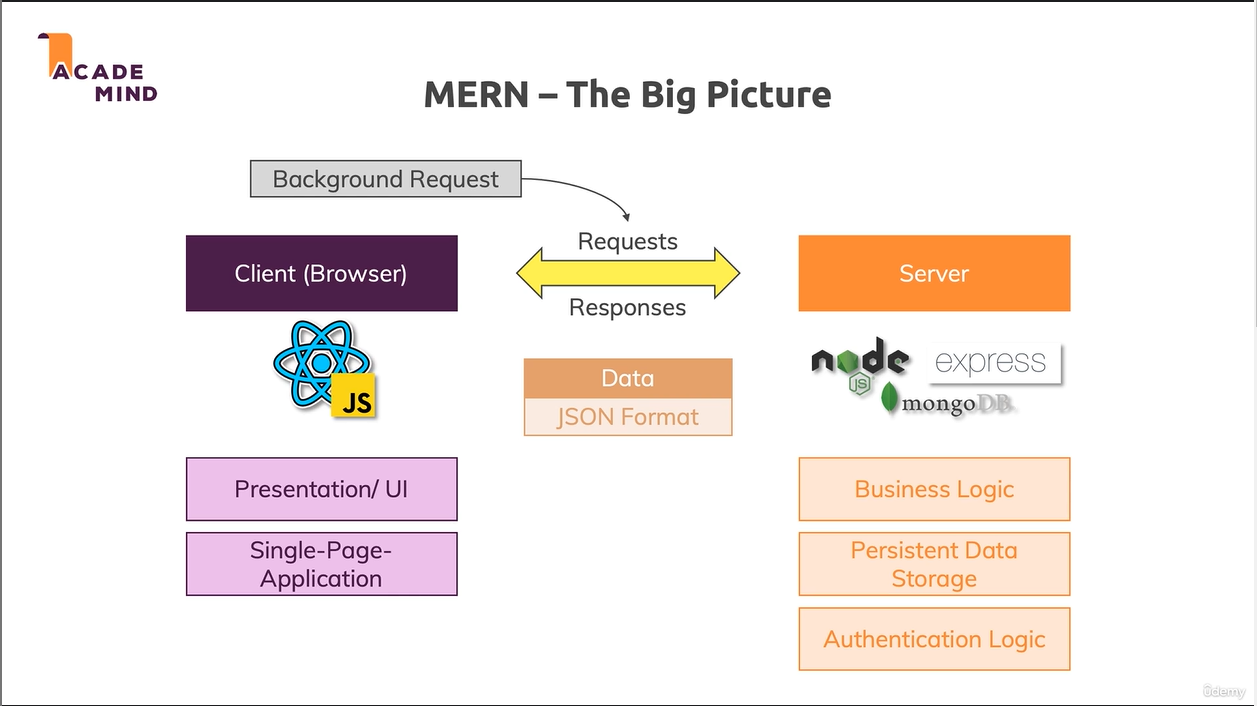
* Based on node, offers same functionalities
* Middleware-based: Funnel Requests through functions
* Includes Routing, view-rendering and more...
* Simplifies the usage of node.

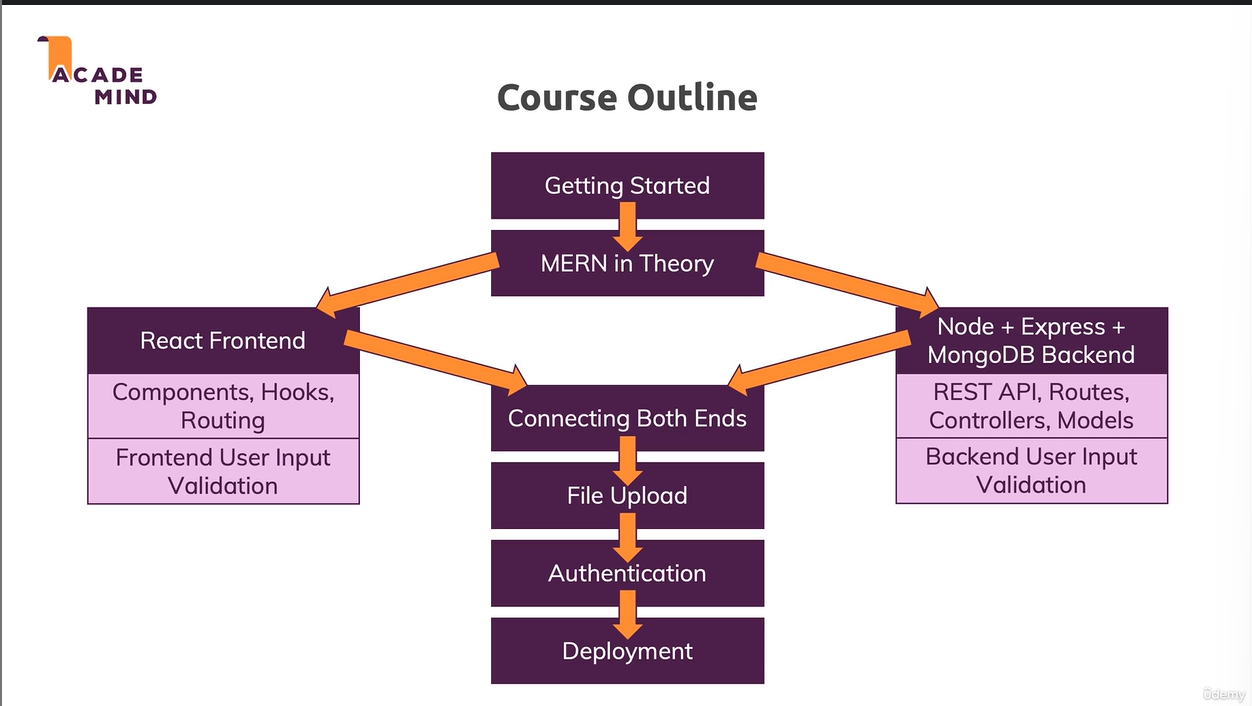
**Mongodb**

A NoSQL database which stores ‘documents’ in ‘collections’ (instead of ‘records’ in ‘tables’ as in SQL.

* Store application data
* Enforces no data schema or relations
* Easily connected to node/express (Not to react)
* A powerful database which can easily be integrated into a node/express environment

**MERN STACK**





**Frontend (Client)**

**React SPA (single page application)**

* Routes (with react-router-dom)
  + Route Config + Page components
* State Management (Hooks, Redux)
  + Redux Logic, React Hooks, Custom Hooks
* Components + Styling (CSS)
  + Utility / UI Components

**Backend (Server)**

**Decoupled Ends**

* Backend == API
  + REST API
    - Different URLs + Http Verbs for different actions
  + GraphQL API
    - One URL + Http Verb that accepts query commands

REST & HTTP Methods

GET – Get a resource from the server

POST – Post a resource to the server (e.g create or append resource)

PUT – Put a resource onto the server (e.g create or overwrite a resource)

PATCH – Update parts of an existing resource on the server

DELETE – Delete a resource on the server

OPTIONS – Determine whether follow-up request is allowed (sent automatically)

|  |  |
| --- | --- |
| REST API | GraphQL API |
| Request path + http method identify a resource / action on the server | Query expression identifies a resource and action |
| API is stateless and decouple from any frontend | API is stateless and decouple from any frontend |
| The most common type of API because of its ease of use | Popular but less common, because you need to learn the query language |

Two ways of connecting Node + React

|  |  |
| --- | --- |
| Server Hosts Node API + React SPA | Two Separated Servers |
| Node (Express) API handles incoming requests | Node (Express) API handles incoming requests |
| Requests not targeting API routes return React SPA | React SPA served from separate static host |
| Data is exchanged between React app and Node API in JSON format | Data is exchanged between React app and Node API in JSON format |

Express 3rd party library

* UUID (generates unique ID)
* body-parser
* express validator (handle validations for API)
* bcryptjs (hashing password)
* jsonwebtoken (create token)

MONGODB

A NoSQL Database stores “Documents” in “Collections” (instead of “Records” in “Tables” as in SQL)

* Store application Data
* Enforces no Data Schema or Relations
* Easily connected to Node/Express
* A powerful database which can easily be integrated into Node/Express Environment

|  |  |
| --- | --- |
| NoSQL | SQL |
| Enforces no Data Schema | Enforces a strict Data Schema |
| Less Focused on Relations | Relations are a Core Feature |
| “Independent Documents” | Records are Related |
| Great for: Logs, Orders, Messages | Great for: Shopping Carts, Contacts, Networks |