

Figure 1 Client Server Model

Servers provide functionalities called services such as sharing data or resources. An API is an example of one such service that Servers provide.

**REST architecture:**

Client-Server Separation

Stateless Constraint

Cache Constraint

Uniform Interface

Node.js provides an environment & foundation for building various applications & Express.js offers a strong open-source community which provides several extensions in the form of middleware.

Middleware are libraries that help our servers work with cookies, sessions, user logins, URL parameters, POST data, security headers & much more.

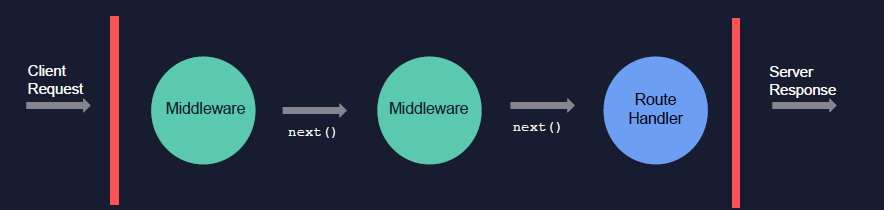


Figure 2 Middleware Functions

Middleware functions handle request response objects before the Routes & perform tasks like Authentication, Data Validation, Body Parsing, Logging & more. By calling next() function, the Request Response object is sent to next Middleware function if present & if not then to the Route handlers.

Routing in Express.js is how an application’s endpoints respond to client requests. Routes can be defined by using the methods the Express app provides. The methods correspond to the HTTP methods.

**A simple Express Route has the following structure:**

app.Method (PATH, HANDLER)

app is the instance of express

method is the HTTP verb (get, post, put, delete)

path is defined path on the server

handler is the callback function executed

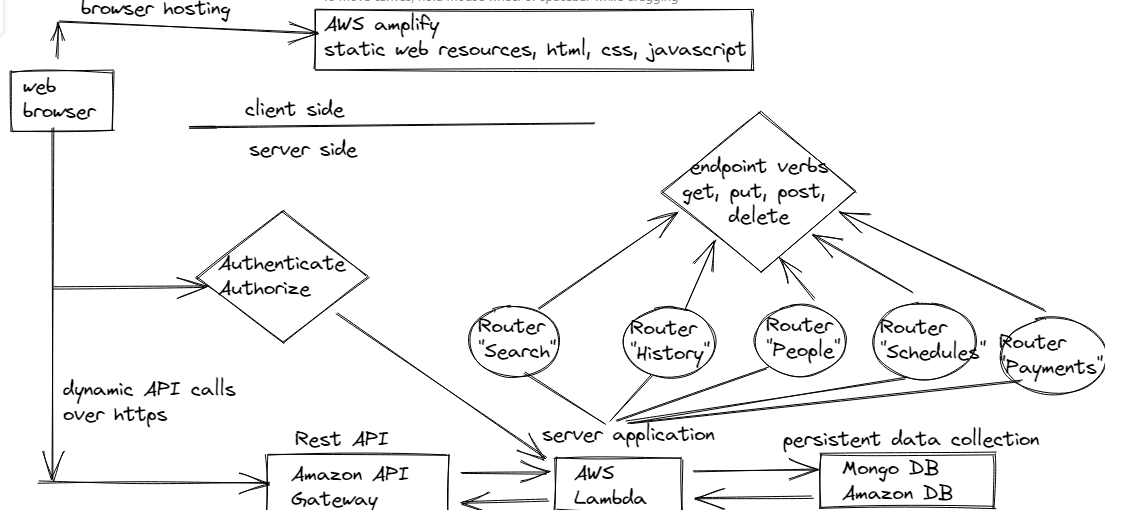


Figure 2. Architecture