

PROJECT TITLE

SHOPPING LIST AND RECIPE DASHBOARD:

Project Description

* A secure SPA Shopping list application that lets the user add a recipe to the dashboard, edit it or delete it. The application has the unique feature of letting the user add the ingredients from a particular recipe to your shopping list.

Goals of the Project

Integrate Front End and Back end and hence develop a proper MEAN stack application

Implement an application that helps with dealing real life scenarios

Implement Security features like using JWT Token, Auth guards, etc.

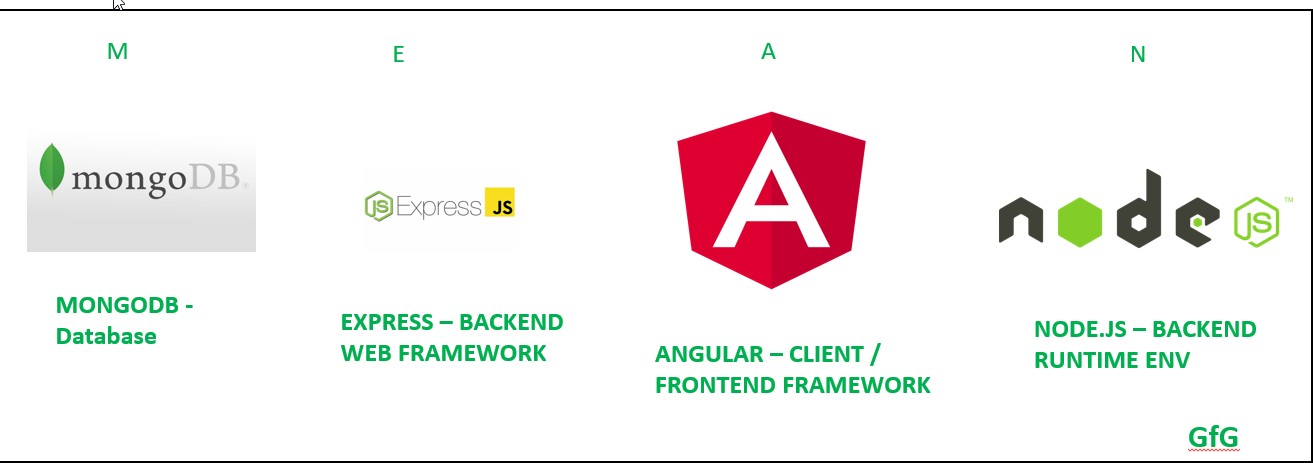
Validation of all user

input forms

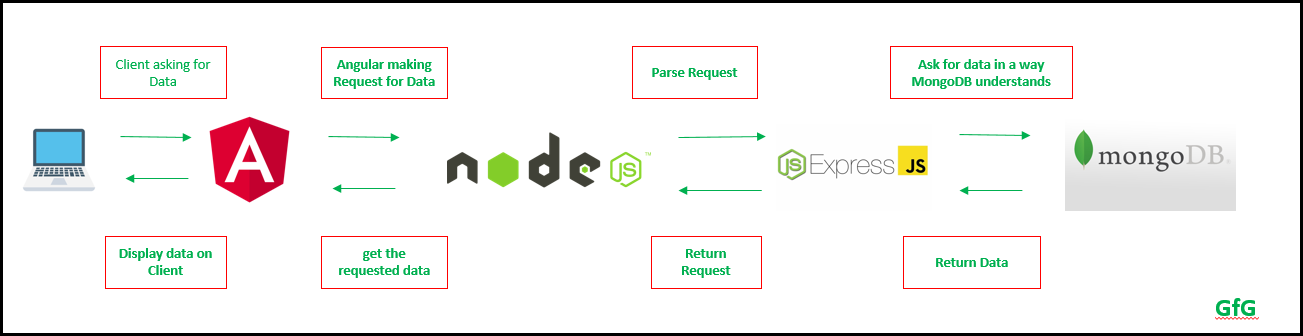
Fetch data from Back

end using API

Tech Stack used



Flow of Data in MEAN Stack Application



* Custom pipes and Directives
* Services for major features
* TypeScript models like a user model, recipe model
* Reactive forms for handling user input with validations
* Responsive
* Shared module for components, directives, and pipes that can be imported into any feature module.

Core concepts implemented:

Frontend: Packages used

Bootstrap :

Sleek, intuitive, and powerful front-end framework for faster and easier web development.

Frontend: Packages used

**RxJS: Reactive Extensions For JavaScript**

RxJS is a library for composing asynchronous and event-based programs by using observable sequences. It provides one core type, the [Observable](https://rxjs.dev/guide/observable), satellite types (Observer, Schedulers, Subjects) and operators inspired

by [Array#extras](https://developer.mozilla.org/en-US/docs/Web/JavaScript/New_in_JavaScript/1.6) (map, filter, reduce,

every, etc) to allow handling

asynchronous events as collections.

Backend: Packages used

* Bcrypt.js: Optimized bcrypt in JavaScript with zero dependencies. Compatible to the C++ [**bcrypt**](https://npmjs.org/package/bcrypt) binding on node.js and also working in the browser. Besides incorporating a salt to protect against rainbow table attacks, bcrypt is an adaptive function: over time, the iteration count can be increased to make it slower, so it remains resistant to brute-force search attacks even with increasing computation power.

Backend: Packages used

* Body-parser: Node.js body parsing middleware.
* Cors: CORS is a node.js package for providing

a [**Connect**](http://www.senchalabs.org/connect/)/[**Express**](http://expressjs.com/) middleware that can be used to enable [**CORS**](http://en.wikipedia.org/wiki/Cross-origin_resource_sharing) with

various options.

* Express: Fast, unopinionated, minimalist web framework for [**node**](http://nodejs.org/).

Backend: Packages used

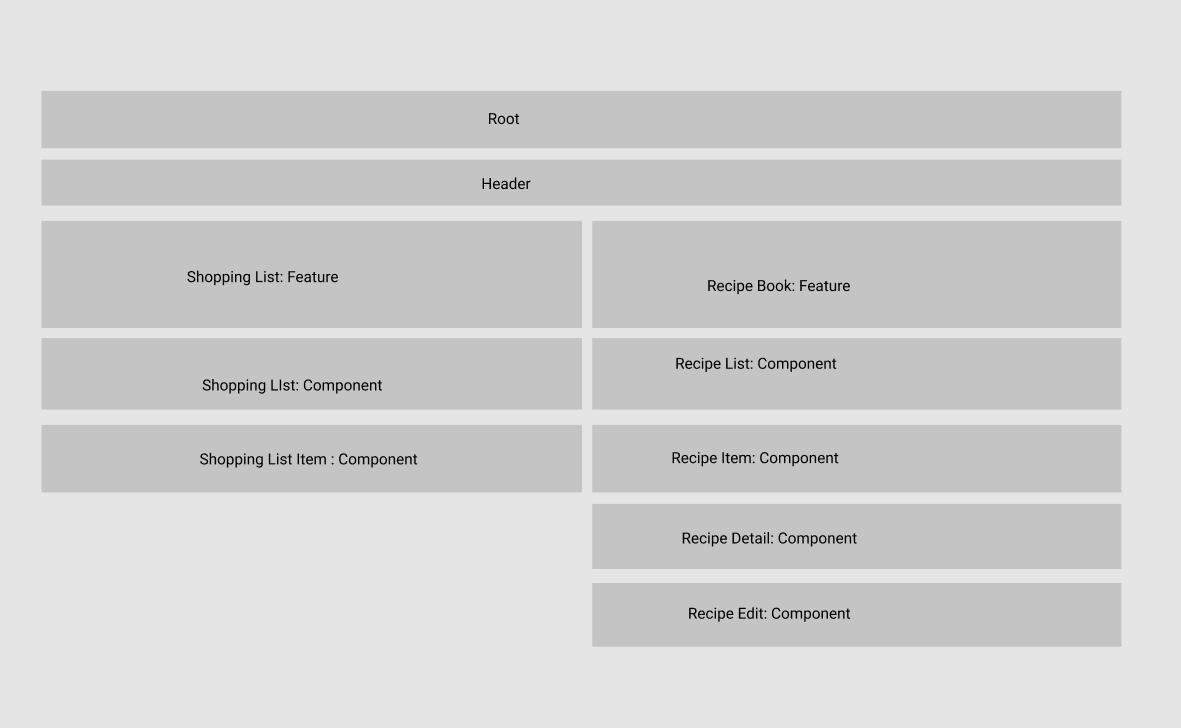
* Jsonwebtoken: An implementation of [**JSON Web Tokens**](https://tools.ietf.org/html/rfc7519).
* Lodash: A modern JavaScript utility library delivering modularity, performance & extras.
* Mongoose: Mongoose is a [**MongoDB**](https://www.mongodb.org/) object modeling tool designed to work in an

asynchronous environment. Mongoose supports both promises and callbacks.

* Passport: Passport is [**Express**](http://expressjs.com/)-compatible authentication middleware for [**Node.js**](http://nodejs.org/).
* Passport-local: Local username and password authentication strategy for Passport



Component Architecture



Thank You