23. Recipe Meal Planner

Creating a Recipe Meal Planner using the MEAN stack (MongoDB, Express.js, Angular, Node.js) is a practical project. Below is a high-level overview of how to build such a tool, along with some code snippets to guide you:

Project Setup and Structure

Set up a new project folder and structure for your Recipe Meal Planner. Install the required Node.js packages and create a basic Angular application.

```
# Create a new Angular application

ng new meal-planner-app
```

- Backend (Node.js & Express.js)

Create the backend of your Recipe Meal Planner using Node.js and Express.js.

Installation of Packages

Install the necessary packages for Express.js, Mongoose (for MongoDB), and other dependencies.

```
npm install express mongoose cors
```

Setting up Express.js

Create your Express.js server, set up middleware, and handle routes.

- **javascript**

```
// server.js

const express = require('express');
const mongoose = require('mongoose');
const cors = require('cors');

const app = express();

// Middleware
app.use(express.json());
app.use(cors());

// Database connection
mongoose.connect('mongodb://localhost/meal-planner-app', {
  useNewUrlParser: true,
  useUnifiedTopology: true,
  useCreateIndex: true,
```

```
});

// Define Mongoose models for User, Recipe, and MealPlan data
const User = mongoose.model('User', {
  username: String,
  password: String, // Use hashing for security
  // Add more user-related fields as needed
});

const Recipe = mongoose.model('Recipe', {
  name: String,
  ingredients: [String],
  instructions: String,
  // Add more recipe-related fields as needed
});

const MealPlan = mongoose.model('MealPlan', {
  userId: mongoose.Schema.Types.ObjectId,
  date: Date,
  recipes: [mongoose.Schema.Types.ObjectId],
  // Add more fields as needed
```

```
});

// Routes for managing users, recipes, and meal plans
app.post('/api/register', async (req, res) => {
  // Register a new user
  // Store hashed password in the database
});

app.post('/api/login', async (req, res) => {
  // Authenticate user and generate a JWT token
});

app.post('/api/recipes', async (req, res) => {
  // Create a new recipe
  // Save the recipe to the database
});

app.get('/api/recipes', async (req, res) => {
  // Retrieve a list of recipes
});
```

// Create similar routes for managing meal plans

- ## Frontend (Angular)

Create the frontend of your Recipe Meal Planner using Angular. Design the user interface for managing recipes, meal planning, and user accounts.

Design and UI

Design the user interface for your Recipe Meal Planner using Angular components, templates, and styles.

Recipe Management

Create components and forms for users to create and manage recipes.

Meal Planning

Design components for users to plan meals by selecting recipes for specific dates.

User Authentication

Implement user registration and login functionality.

- **typescript**

// recipe-management.component.ts

```
import { Component } from '@angular/core';
import { RecipeService } from './recipe.service';


@Component({
  selector: 'app-recipe-management',
  templateUrl: './recipe-management.component.html',
})
export class RecipeManagementComponent {
  name: string;
  ingredients: string[];
  instructions: string;

  constructor(private recipeService: RecipeService) {}

  createRecipe() {
    this.recipeService.createRecipe(this.name, this.ingredients,
this.instructions);
  }
}
```

MongoDB

Create a MongoDB database to store user profiles, recipes, and meal plans.

Putting It All Together

Integrate the frontend and backend by making API requests from Angular components to Node.js routes. Ensure that you handle recipe creation, meal planning, user authentication, and meal plan storage properly.

Building a Recipe Meal Planner is a practical project that can be expanded with features like meal shopping lists, nutritional analysis, user ratings for recipes, and automated grocery list generation. It's a versatile tool that can be customized to your specific requirements.