21. Task Management App

Building a Task Automation tool using the MEAN stack (MongoDB, Express.js, Angular, Node.js) can greatly enhance productivity and efficiency. 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 Task Automation tool. Install the required Node.js packages and create a basic Angular application.

```
# Create a new Angular application
ng new task-automation-app
```

- ## Backend (Node.js & Express.js)

Create the backend of your Task Automation tool 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/task-automation-app', {
  useNewUrlParser: true,
  useUnifiedTopology: true,
  useCreateIndex: true,
});

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

const Task = mongoose.model('Task', {
  title: String,
  description: String,
  deadline: Date,
  completed: Boolean,
  // Add more fields as needed
});

// Routes for managing users and tasks
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/tasks', async (req, res) => {
  // Create a new task
  // Save the task to the database
});

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

// Create similar routes for managing tasks
```

- Frontend (Angular)

Create the frontend of your Task Automation tool using Angular. Design the user interface for managing tasks and user accounts.

Design and UI

Design the user interface for your Task Automation tool using Angular components, templates, and styles.

Task Management

Create components and forms for users to create, update, and manage tasks.

User Authentication

Implement user registration and login functionality.

- **typescript**

```
// task-management.component.ts
import { Component } from '@angular/core';
import { TaskService } from './task.service';

@Component({
  selector: 'app-task-management',
  templateUrl: './task-management.component.html',
})
export class TaskManagementComponent {
  title: string;
  description: string;
  deadline: Date;

  constructor(private taskService: TaskService) {}

  createTask() {
    this.taskService.createTask(this.title, this.description, this.deadline);
  }
}
```

MongoDB

Create a MongoDB database to store user profiles and task data.

Putting It All Together

Integrate the frontend and backend by making API requests from Angular components to Node.js routes. Ensure that you handle task creation, management, and user authentication properly.

Building a Task Automation tool is a practical and highly customizable project. You can expand it with features like task categorization, task scheduling, notifications, and reporting for a more sophisticated automation experience.