12. Calendar Scheduler

Creating a Calendar Scheduler application using the MEAN stack (MongoDB, Express.js, Angular, Node.js) is a comprehensive project. Here's a high-level overview and some code snippets to guide you:

Project Setup and Structure

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

```
# Create a new Angular application
ng new calendar-scheduler
```

- Backend (Node.js & Express.js)

Create the backend of your Calendar Scheduler application 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/calendar-scheduler', {
  useNewUrlParser: true,
  useUnifiedTopology: true,
  useCreateIndex: true,
});
```

```
// Define Mongoose models for Event data
const Event = mongoose.model('Event', {
  title: String,
  description: String,
  start: Date,
  end: Date,
  location: String,
  // Add more fields as needed
});

// Routes for managing events
app.post('/api/events', async (req, res) => {
  // Create a new event
  // Save the event to the database
});

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

app.put('/api/events/:id', async (req, res) => {
```

```
  // Update an event
  // Save the updated event to the database
});

app.delete('/api/events/:id', async (req, res) => {
  // Delete an event
  // Remove the event from the database
});
```

- ## Frontend (Angular)

Create the frontend of your Calendar Scheduler using Angular. Design the user interface for scheduling events and appointments.

Design and UI

Design the user interface for your Calendar Scheduler using Angular components, templates, and styles.

Event Scheduling

Create components and forms for users to schedule and manage events, including specifying the title, description, date, time, and location.

- **typescript**

```
// event-schedule.component.ts
import { Component } from '@angular/core';
import { EventService } from './event.service';

@Component({
  selector: 'app-event-schedule',
  templateUrl: './event-schedule.component.html',
})
export class EventScheduleComponent {
  title: string;
  descriptionv: string;
  start: Date;
  end: Date;
  location: string;

  constructor(private eventService: EventService) {}

  scheduleEvent() {
    this.eventService.scheduleEvent(this.title, this.description,
this.start, this.end, this.location);
  }
}
```

MongoDB

Create a MongoDB database to store event 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 event scheduling, updating, and deletion properly.

Building a Calendar Scheduler is a practical project, and you can enhance it with additional features like event reminders, recurring events, and notifications. Consider using external libraries for calendar views and scheduling interfaces to provide a better user experience.