Admin Panel Specification

- CGD -

The website for CGD is done and the focus of yours will be to create an application that will implement the requirements for the Admin panel of CGD.
The Admin panel will be a separate application i.e. it won't be in the same codebase as the website, which means that this application you will be working on will consist only of the code you will write.

The idea of this Admin panel application is to implement all CRUD operations for all the data that the website uses and to eventually serve the website by integrating the two of them.
The code you will write will be only focused on the functionalities of the Admin panel itself and the scope of this project doesn't include taking care of the integration of the Admin panel with the website.

The functionalities that this Admin panel should provide are listed and explained below.

Functionalities

1. **Jobs:**
    - CRUD for jobs with the following data fields:
        - **Title** of the job position *(title, string)*
        - **Job description** *(description, text)*
        - **Job type** (Part time, Full time) *(type, string)*
        - **Work mode** (Hybrid, On-site) *(work_mode, string)*
        - **Location** *(location, string)*

2. Team members

- CRUD for Team members with the following data fields:
    - **Picture** - URL of a picture *(picture, string)*
    - **Name** *(name, string)*
    - **Surname** (surname, string)
    - **Position** *(position_id, foreign key to **positions** table)*
    - **Short profile** *(short_profile, text)*

3. Partners

- CRUD for Partners with the following data fields:
    - **Name of the company** *(company_name, string)*
    - **Industry** *(industry_id, foreign key to **industries** table)*
    - **Company Logo** - URL of a picture *(logo, string)*
    - **Collaboration description** *(collaboration_description, text)*

4. Testimonials

- CRUD for Testimonials with the following data fields:
    - **Testimonial Text** *(testimonial_text, text)*
    - **Client Name** *(client_name, string)*
    - **Client Position** *(client_position, string)*
    - **Client Company** *(client_company, string)*
    - **Client Profile Picture** - URL of a picture *(client_profile_picture, string)*

5. Contact Form handler

On the website there is a contact form that we want to handle in this app. Namely, whenever someone sends a message from the contact form on the website, that form would submit to an API endpoint that will be implemented within this (your) app.
So whenever this API endpoint is triggered i.e. whenever someone submits the form on the website you should handle that request by implementing the following things in the API endpoint:

- Save the subject, message and the email address from the user who sent the message;
- Send an automatic email letting the user know that their message was received and they will get an answer very soon; **You should send the email using a Laravel Event;**

Note: *There is no need to implement an authentication system for the API endpoint.*

6. Industries

- CRUD for Industries with the following data fields:
  - **Industry Name** *(name, string)*
  - **Description** *(description, text)*
  - **Industry icon** - URL of a picture *(icon, string)*

7. Services

- CRUD for Services with the following data fields:
  - **Service name** *(name, string)*
  - **Service description** *(description, text)*
  - **Service category** *(service_category_id, foreign key to **service_categories** table)*
  - **Service industry** *(industry_id, foreign key to **industries** table)*

8. Service categories

- We need only the **CREATE/STORE** functionality for Service categories, with the following data field:
  - **Name** *(name, string)*

9. Authentication and User management

The Auth system can be implemented by using the default Laravel authentication system provided by Breeze;

The **users** *table* will have the following data fields:

- **Username** *(username, string)*
- **Role** *(role_id, foreign key to roles table, default value: admin)*
- **Email Address** *(email, string)*
- **Password** *(email, string - **has to be*** **hashed***)*

Additionally you will need to implement **Roles** table that will be seeded with only *one role* - ***admin***;

There ***won't be*** any User CRUD operations, you only need to add **one user** in the database using a seeder and the following data:

- **Username**: *admin*
- **Role**: *admin*
- **Password**: *admin*
- **Email address**: *admin@cgd.com*

10. Dashboard

The Dashboard is the first thing the users will see after logging in;

This screen should consist of a sidebar or a navbar inside of which you will add links to the Lists of all the resources (a link to the list of Jobs, another link to the list of Team members, then Services, Industries, etc.)

Aside from this sidebar/navbar, the content of this page should consist of the following 2 sections:

- **A statistics chart for jobs published over the previous year by month** *(X-axis should be the months of a year, Y-axis should be the number of jobs)*;

    - **Preview of messages received from the Contact form;**
        - Always list the last 5 messages (in this list you should show only the subject of the message, not the full message);
        - If there are new messages i.e. messages that are unread you should show a red-coloured badge with the number of new messages in it; An unread message is one that was never opened before. Any message is unread by default and it changes the status to "*read*" when it's opened for the first time.
        - From the list of messages they should be able to see the full message by clicking on it, and on the next screen where the full message can be seen the user should be able to remove them, which removes them from the database using **Soft Delete** technique;

Clarifications:

➢ **CRUD** means that we need full functionality for **Create/Store, Read/Show, Edit/Update** and **Delete** operations.
  - The **READ** functionality should be the same for all the resources. You need to show a screen with a table that consists of all the information of the corresponding database table.
  - The **CREATE** & **EDIT** operations should consist of a screen with a form which will submit to the corresponding routes.
    To be able to get to the **EDIT** form of a particular item, we should have an **EDIT** button in the table where we can see all the items.
  - The **DELETE** functionality can be the same for all of the resources as well. In the table where all the data of the resource will be visible we should have a **DELETE** button which will delete the corresponding resource from the database.

➢ Regarding the Contact form, since you won't have an integration with the website at this time, in order to be able to showcase the preview of messages on the dashboard, you will need to add some messages by using a seeder where you'll add both read and unread messages; The functionality of the API endpoint will be graded only by the code implementation;

➢ Regarding the Statistics chart, you will also need to add some jobs using a seeder; You can use any JS library in order to implement the chart;

➢ Regarding the design of this project, you have all the freedom to use your imagination and implement it the way you want; Simple bootstrap design will do too; The design won't be part of the scoring system for this project, though it's expected from you not to deliver default HTML elements design, but use Bootstrap at the very least;