Cura Dot Net Backend Developer Challenge

Introduction

- The main purpose of this challenge is to let you show off your personal software development skills and abilities so that we can see how you think and how you approach software design and implementation.

Guidelines:

1. Please feel free to pick any modern technologies and any readymade or 3rd party plugins and controls if you need to.
2. Please choose **only one of the below 3 assignments** to apply.
3. If you have technical questions and feel stuck about your task, just make an assumption and write a comment to clarify your assumption and proceed. The tasks were written deliberately in a non-specific way to allow for room for your imagination, assumptions and design skills.

Task Deliverables:

1. A hyperlink to a GitHub or Bitbucket repo or any other platform of your choice that you uploaded the code to.
2. If you want to clarify something or explain other stuff, feel free to write that in your reply email.

Assignment #1 - Designing maintainable and extensible code:

Scenario:
You are an advocate of clean code and maintainable design, you are tasked to create an organization level Notification Service for a breaking news agency to be used in multiple systems.

Task:

- Develop a notifications service (Email and Push notifications), the service must be extensible with support for future notification providers such as SMS notifications, Slack Notifications, and others.
- The plugin shall be developed as a C# class library and you can use Pseudo code for SMTP, SMS API, etc.. just make sure that the code is clear, and we can understand the flow you have in mind.
- Keep in mind that this service will be used as a plugin by other developers in your organization, on multiple and varied projects, try your best to make it extensible, compatible, and developer friendly.
- The targeting (input) data should be passed as parameters, and the function shall return a single/list of Notification Object/s that contain\s a GUID as the notification Id and have been created per the request.

- Clean code and design patterns will heavily influence the submission's rating, we suggest using the Decorator pattern, but feel free to impress us if you have a better approach.
- The system shall implement a non-blocking logic path and might have an isolated database (up to you), but an important thing to note is that the class library doesn't have any access to application database and shall be designed as an external library.

Assignment #2 - Create a custom caching layer:

Scenario:
You have a clustered (Load balanced) system, your system is doing complex operations of the kind you only see on University Calculus exams, alongside process heavy data querying.
A dear System Architect friend of yours suggested that you implement centralized caching on requests, where you would cache responses on requests to serve them from the centralized cache.

Task:
- Create a custom caching layer for a Web API (.Net Framework or .Net Core).
- Use MSSQL for storing the responses.
- Keep in mind that request parameters do play a role what are the responses, and cached items are considered not fresh after 2 hours of their creation and need to be deleted from the Database.
- Consider the API system is stateless and only request parameters are to be checked against.
- Do not waste your time on non-important sections of the system, we will only evaluate the caching layer for performance, maintainability, correctness criteria, flexibility, and edge case handling.

Assignment #3 - Authentication API:

Scenario:
You are working with an amazing team that develops a super popular mobile game, the team decided to make the game only available to logged in users.

Task:
- Design an oAuth login API, the API shall tell the user his/her roles and what systems regions he/she can access.
  System example regions:
  1. Board game "b_game" (default for all logged in users)
  2. VIP Character modification "vip_chararacter_personalize" "For VIP users"

System roles example:

1. Player “player”
2. Administrator “admin”

- Kindly develop this API in ASP.Net MVC or .Net Core MVC (select one) and show us how you would structure the JWT token.
- You need to have a working API (user data can be static), with username and password authentication.
- The JWT token must contain the roles and scope within the claims section.
- You can skip Database design, develop against static data.