.NET Technical Test

Blog Engine .NET APP

v3.0 Febrero 2021

OVERVIEW

Build a minimal Blog Engine / CMS backend API, that allows to create, edit and publish text-based posts, with an approval flow where two different user types may interact.

SPECIFICATIONS

PART 1 - Mandatory

Build a RESTful API for a blog engine using .NET.

The API should use some authentication mechanism to identify valid users and authorize actions based on the user's role. The roles for the application are: "Public", "Writer" and "Editor".

The API should allow the following operations for the specified roles:

- Retrieve a list of all published posts (all roles)
- Add comments to a published post (all roles)
- Get own posts, create and edit posts (Writer)
  - Writers should be able to create new posts and retrieve the posts they have written.
  - Writers should be able to edit existing posts that haven't been published or submitted.
- Submit posts (Writer)
  - When a Writer submits a post, the post should move to a "pending approval" status where it's locked and cannot be updated.
- Get, Approve or Reject pending posts (Editor)
  - Editors should be able to query for "pending" posts and approve or reject their publishing. Once an Editor approves a post, it will be published and visible to all roles. If the post is rejected, it will be unlocked and editable again by Writers.
  - Editor should be able to include a comment when rejecting a post, and this comment should be visible to the Writer only.

Each post must include a title, some text content, the date of publishing and its author.

PART 2 - Bonus Points

Build a UI web application to interact with the API. The UI should, as a minimum, display a list of all published posts, and the full contents (including comments) of any particular post when selected.

We won't be evaluating the UI/Design side of things. Your focus must be on functionality and correct integration with the API, but feel free to use and show off your Front-End skills.

You can use whatever Web development stack you're most comfortable with (ASP.NET MVC/Razor pages, .NET Blazor, JS-based frameworks such as React, Vue, Angular, etc.).

Design Rules

- You can use the API framework of your choice (ASP.NET, Serverless Functions, ServiceStack, etc.)
- You can use the Storage solution of your choice (SQL database with EF or other ORM, NoSQL data stores, flat files, etc.)
- You must use a Dependency Injection/IoC container of your choice.
- Login storage can be as simple as hardcoded users/passwords, but the api must use an authentication mechanism to secure its operations.
- All requests and responses must be in JSON format.

DELIVERABLES

Required

- Complete source code and script of database (if applicable). Preferred option is a public repository on **github**, **gitlab** or **bitbucket**.
  If you're not able to publish to a public repository then please use git on your local machine and bundle the zip to us (https://git-scm.com/book/en/v2/Git-Tools-Bundling).
- README file with the detailed steps to get the application up and running: software prerequisites, steps to build the app, etc, as well as the total time (in hours) it took to complete the test.
- Sample credentials for different types of user.

Optional (show off your skills!)

- Postman collections OR curl commands to test the API operations.
- Swagger definition for the API.
- Relevant Unit Tests covering the business logic in the API.
- If you have experience and access to Cloud Platforms, deploy the solution to it so it can be tested without local install. You can deploy to Azure/AWS/GCP, etc.