POC Task — Online Skill Exchange Portal (Blazor + .NET 8)

Objective

Build an **Online Skill Exchange Portal** to practice full-stack development with Blazor frontend and .NET 8 backend. The purpose is to evaluate your ability to design and implement role-based functionality, content workflows, and integrations (file storage).

Tech Stack & Guidelines

- Frontend: Blazor with **MudBlazor** components for responsive UI.
- Backend: .NET 8 Web API.
- Architecture: **API (Controller) → BAL (Manager with Interface) → DAL (Repository with Interface)**.
- ORM: Entity Framework Core (code-first) for Insert/Update/Delete operations (use LINQ).
- Reads & Searches: Use **Stored Procedures** for all GET / search operations.
- DB: SQL Server.
- Authentication: JWT or similar auth with Admin / User roles.
- File Storage: Upload all content (videos, images, PPTs) to **AWS S3 bucket**.

Roles & Features

User - Self-register with account details. - Until admin verification, account shows as **Under Verification**. - During verification period, can send messages to Admin. - Once verified, can login and access full portal features. - Explore content by categories/skills. - Search/filter content grid by stars, reviews, or titles. - Choose any course/skill learning program and consume the content (video, image, PPT). - Provide optional feedback at the end of a course. - Upload new content based on their own skills. Uploaded content goes for **Admin approval** before being public.

Admin - Verify newly registered user accounts. - Approve/reject uploaded content before it becomes visible to all users. - Manage categories/skills for classification. - View user feedback on courses.

Functional Requirements

- User registration and verification workflow.
- Content browsing by categories/skills, with search & filter (stars, reviews, titles).
- Content consumption (video, image, PPT).
- Feedback submission (optional).
- Content upload by users with S3 storage integration.
- Admin approval workflow for new content.

- Admin verification workflow for new accounts.

Non-Functional Requirements

- Responsive UI using MudBlazor.
- Clear separation with API → BAL → DAL architecture.
- Stored procedures for search and list queries.
- EF Core (LINQ) for insert/update/delete.
- Proper validations on forms and uploads.
- Logging and error handling.

Deliverables

1. Source code (frontend & backend) in Git repository with clear README.
2. SQL migration scripts and stored procedure scripts.
3. Postman collection or API documentation.
4. Demo steps or short recording showing: registration, admin verification, exploring content, uploading content, and admin approval.
5. Brief write-up on architecture choices and AWS S3 integration.

Evaluation Checklist

- Correct use of Blazor + MudBlazor for frontend and .NET 8 for backend.
- Stored procedures for read/search, EF Core (LINQ) for CUD operations.
- Clean API → BAL → DAL architecture.
- Role-based security and workflows for User and Admin.
- Correct integration with AWS S3 for content upload and retrieval.
- Responsive UI and user experience.
- Clear documentation and ability to run locally.

End of Task Definition