Telerik Academy

Poodle

e-Learning Platform

Project Description

Your task is to design an e-learning platform that would be used by students to search for and enroll in online courses and teachers who would publish the courses.

On high level, the system would consist of three main entities:

Users (either teacher or student), Courses (either public or premium), Sections**.**

Students must be able to view available courses and access their content depending on whether the courses are public or premium and/or they have subscribed for them or not.

Teachers must be able to create and update courses, add, edit, and remove sections for them.

Admins could authorize teachers' registrations, deactivate access for students, hide / delete courses. The admin role comes with predefined log-in details where only the password can be amended (could).

Functional Requirements

- Each **user** must have an email, first name, last name, and a password.
- The email serves as a username for both **students** and **teachers** and cannot be amended).
- **Teachers** should also have a phone number and linked-in account.

- Each **course** must have a title (unique), description, objectives, owner (teacher), tags with relevant expertise areas and sections.
- Each **course** should optionally have a Home Page picture.
- Each **course** must be either public or premium and can later be augmented with sections.
- Each **course** should have an option for subscribing or unsubscribing.
- Each **course** could have a rating which represents a proportionate value of the provided scores. As an example if 2 people rate a course with 7 out of 10 and 6 out of 10, then the calculated rating would be 7 plus 6 divided by 20 (7 + 6 = 13 / 20 = 0.65). The rating would be **6.5 out of 10.**

- Each **section** must have a title, content type, description (optional), information / link to external resource(optional).
- **Sections** within a course should have an option to be sorted by id or name**.**

Public Part

The public part must be accessible without authentication i.e., for anonymous users.

- **Anonymous users** must be able to view the title and description, tags of available public courses, but not be able to open them.
- **Anonymous users** could search courses by tag and/or rating.
- **Anonymous users** must be able to register.
- **Anonymous users** must be able to login.

Endpoints for registered users

Accessible only if the user is authenticated. If the user is authenticated as a teacher or admin (could), they would be able to access all courses and sections. If logged in as a student, they would access all public courses and only the premium courses in which they are enrolled.

For Students

- **Students** must be able to view and edit their account information (except the username).
- **Students** should be able to track their progress for every course based on the sections that they have visited (i.e. if a course has **7** Sections and the Student visited **4** of them the progress would be **57%)**
- **Students** must be able to view the courses that they are enrolled in (both public and premium).
- **Students** must be able to view and search through by name and tag existing public and premium courses.
- **Students** must be able to unsubscribe from premium courses.
- **Students** should be able to rate a course (only one score for a course) only if they are enrolled in it.
- **Students** should be able to subscribe to a maximum of 5 premium courses at a time and unlimited number of public courses.

For Teachers

- **Teachers** must be able to view and edit their account information (except the username).
- **Teachers** must be able to create courses, view and update their own courses.
- **Teachers** could be notified via email whenever an enrollment request is sent by a student for a specific course that they own.
- **Teachers** should be able to approve enrollment requests sent by students.
- **Teachers** could be able to deactivate / hide only courses to which they are owners when there are no students subscribed for that course.
- **Teachers** should be able run report for the past and current students that have subscribed for their courses.

For admins (could)

- **Admins** could approve registrations for teachers.
- **Admins** could be able to view a list with all public and premium courses, the number of students in them and their rating.
- **Admins** could deactivate/reactivate students.
- **Admins** could delete / hide courses and the enrolled students could receive a notification that the course is no longer active.
- **Admins** could remove students from courses.
- **Admins** could search through courses filtered by teacher and/or by student.
- **Admins** could trace back ratings for courses to identify the students who scored the course.

Optional features

- **Search endpoints** should support pagination and sorting.

- **Email Verification** for the teachers could be implemented. In order for the registration to be completed, the teacher must verify their email by clicking on a link send to their email by the application. Before verifying their email, teachers cannot create courses.
- **Email notifications** could be supported (i.e. with the help with a third party service like: https://dev.mailjet.com/email/guides/send-api-v31/.)
- Add **Easter eggs** whenever you could**.** Creativity is always welcome and appreciated. Find a way to add something fun and/or interesting, maybe an Easter egg or two to your project to add some variety.

Technical Requirements

General

- Follow KISS, SOLID, DRY principles when coding
- Follow REST API design best practices when designing the REST API (see Appendix)
- Use tiered project structure (separate the application in layers)
- The service layer (i.e., "business" functionality) must have at least 80% unit test code coverage
- You should implement proper exception handling and propagation
- Try to think ahead. When developing something, think – "How hard would it be to change/modify this later?"

Database

The data of the application must be stored in a relational database. You need to identify the core domain objects and model their relationships accordingly. Database structure should avoid data duplication and empty data (normalize your database).

Your repository must include two scripts – one to create the database and one to fill it with data.

Git

Commits in the GitLab repository should give a good overview of how the project was developed, which features were created first and the people who contributed. Contributions from all team members must be evident through the git commit

history! The repository must contain the complete application source code and any scripts (database scripts, for example).

Provide a link to a GitLab repository with the following information in the README.md file:

- Project description
- Link to the Swagger documentation
- Link to the hosted project (if hosted online)
- Instructions how to setup and run the project locally
- Images of the database relations (must)

Optional Requirements

Besides all requirements marked as should and could, here are some more *optional* requirements:

- Integrate your project with a Continuous Integration server (e.g., GitLab's own) and configure your unit tests to run on each commit to your master branch
- Host your application's backend in a public hosting provider of your choice (e.g., AWS, Azure, Heroku)
- Use branches while working with Git

Teamwork Guidelines

Please see the Teamwork Guidelines document.

Appendix

- Guidelines for designing good REST API
- Guidelines for URL encoding
- Git commits - an effective style guide
- How to Write a Git Commit Message

Legend

- Must – Implement these first.
- Should – if you have time left, try to implement these.
- Could – only if you are ready with everything else give these a go.