Telerik Academy

Social Network

Team Project Assignment

.NET Team Project

This document describes a **team project assignment** for the **.NET cohort** at Telerik Academy.

Project Description

Your task is to develop a Social Network web application that enables users to:

- Connect with people
- Create, comment and like posts
- Get a feed of the newest/most relevant posts of your connections.

The exact theme of the application is intentionally not specified and is limited only to your imagination and abilities. It could be inspired by Facebook and have its focus on the feed generation or be targeted to travelers and have its posts in the format of reviews or be a place to keep in touch with you fellow students.

Functional Requirements

Public Part (anonymous users)

The public part of your projects should be **visible without authentication**. This includes a home page with links to:

- Login form
    - (optional) forgotten password
- Registration form
    - (optional) email confirmation, profile search and public feed
- Search profiles – search for profiles based on name and/or email
    - (optional) by public info
- Public feed – chronologically ordered public posts.
- Public profile – shows the public information associated with a profile.

Private Part (authenticated users)

Accessible after **successful login**.

Update your profile

Once you are logged in, you should be able to change your profile information by:

- Updating your name, email, and any other additional information (date of birth, nationality, job title etc.)
- Uploading a profile photo
- Setting the visibility of your profile photo to:
    - Public - visible to everyone, including anonymous users. *This is the default setting.*
    - Friends - visible only to your friends
    - (optional) Friends except… - do not show to some friends
    - (optional) Specific friends – only show to some friends
    - (optional) Only me
- (optional) Changing your password

Add a friend
You can connect with another user by sending a friend request. Friend requests need to be approved by the other user before they become active.

Remove a friend
You can remove a user from your list of friends. This does not require approval by the other user.

Create a post
As a registered user you can create a post. In its most basic form, a post must contain some text, but ideally you should be able to attach a photo, video, or location

Additionally, you can set the visibility of the post to:

- Public - visible to everyone, including anonymous users
- Friends - visible only to your friends. *This is the default setting.*

Like a friend’s post
As a registered user you can like the posts of your friends. Subsequently you can unlike any previously liked posts. The total number of likes a post generates should also be visible.

- (optional) Include the comment like count in the feed generation algorithm

Comment on a friend’s post
You can comment on a friend’s post by using the comment section underneath the post. Another section within the post shows **3** of the latest comments. When expanded, older comments are loaded.

- (optional) Include comment reply functionality

News Feed

There is a personalized news feed comprised of your friends' posts. The most basic requirement is that the feed is generated by listing all of your friends' posts in chronological order. However, a more complex algorithm should be considered. For example:

- (optional) You can use the number of interactions with the post (comments, likes etc.) and place posts with more interactions higher in the feed
- (optional) You can generate the feed based on user location, and show higher in the feed posts, that are tagged closer to the user

Apply the feed generation algorithm on the public feed section.

Administrative Part

System administrators should have administrative access to the system and permissions to administer all major entities across the system. Administrators can:

- Edit/Delete profiles
- Edit/Delete post
- Edit/Delete comments

REST API

Provide a REST API that leverages HTTP as a transport protocol and clear text JSON for the request and response payloads.

Document your REST API using Swagger.

Be mindful of what you want to provide through your REST API and do not go blindly exposing the entire service layer. Nevertheless, here is a list of endpoints that must be supported.

- Get a user's profile information
- Get list of friends
- Get newsfeed posts
- Create post

Note: All privacy rules apply for REST API endpoints. For example, an unauthenticated user cannot get a list of friends.

Development Guidelines

Your application should use the following technologies, frameworks, and development techniques:

- Use **C#** and the **latest stable versions of the .NET Core SDK**, **ASP.NET Core MVC** and **Visual Studio Community Edition**
- Use **Microsoft SQL Server** as database provider, Microsoft SQL Server Managements Studio and Microsoft Server SQL Profiler for any additional database-related tasks
- Use **Entity Framework Core** to access your database
- Use the default **ASP.NET Core Identity System** for managing users and roles
- Create and use a **service layer** (business functionality) with **at least 80%-unit test code coverage**
    - Use **MSTest** (unit testing framework), **Moq** (mocking framework) and **in-memory database provider**
    - Use the **Dependency Injection** technique and follow the best practices you have learned so far
- Use both **server-side and client-side validation**
- Use **AJAX requests** as much as possible in order to improve the overall user experience of your application
- Use **Bootstrap** to create beautiful and responsive UI
- Use **GitLab** for source control
    - Use feature **branches** during development
    - (optional) Use continuous integration service
- **Documentation** of the project and project architecture (as .md file, including screenshots, diagrams, etc.)

- (optional) Host your application on **Azure**
- Follow **OOP principles** when coding
- Follow KISS, SOLID, DRY principles when coding
- Follow best practices when designing your REST API

Deliverables

- Provide a **link to** your **GitLab** repository and **Trello** board.
    - Keep in mind that commits in the GitLab repository should give a good overview of how the project was developed, which features were created first etc. and the

people who contributed. Contributions from all team members MUST be evident through the git commit history!
    - Read https://dev.to/pavlosisaris/git-commits-an-effective-style-guide-2kkn and https://chris.beams.io/posts/git-commit/ for a guide to write good commit messages

- The repository MUST contain the complete application source code.
- The project MUST have a **README.md** file that complies to the requirements outlined in the previous section.

Project Defense

Each team will have a **public defense** of their work to the trainers and students. It includes a live **demonstration** of the developed web application (please prepare adequate sample data). Also, each team will have a defense of their project with the trainers where they must explain the application structure, major architectural components and selected source code pieces demonstrating the implementation of key features.

Expectations

You MUST understand the system you have created.

Any defects or incomplete functionality MUST be properly documented and secured.

It is OK if your application has flaws or is missing one or two MUST's. What's not OK is if you do not know what is working and what is not and if you present an incomplete project as functional.

Some things you need to be able to explain during your project defense:

- What are the most important things you have learned while working on this project?
- What are the worst "hacks" in the project, or where do you think it needs improvement?
- What would you do differently if you were implementing the system again?

Appendix

1. Guidelines for designing good REST API
2. Guidelines for URL encoding
3. Always prefer constructor injection

4. Git issue tracking docs
5. Gitflow Workflow
6. Git commits: An Effective Style Guide
7. How to write a Git Commit message