Project Assignment for the Web Design and Development course @ UBT

Design and implement a front-end web application **using front-end frameworks with Web APIs or Semantic Web tech backend**.

- **Front-end**
  - Use **JavaScript, Typescript, jQuery or AJAX** the major part of your work should be JavaScript/Typescript written by your team.
  - Any framework is permitted: React, Angular, Vue, mobile UI frameworks, etc.
- **Backend**
  - (Option 1) **Use Web APIs architectural style** for different application services
    - Use at least two Web services with corresponding endpoints
  - (Option 2) **Use Semantic Web technologies** for data storage, retrieval and manipulation
    - Define at list one RDF store for storing data
    - Use at least five SPARQL queries for querying data on RDF stores
    - (Advanced) Define CRUD operations on RDF store for each data entity

Work in team of **exactly 3 persons**. Your project must meet the requirements listed below.

General Requirements

- Use **HTML5, CSS3 or a CSS framework** to create the content and style your Web application.
- **User Interface, Usability, UX and browser support**
  - Design a responsive and user-friendly easy-to-use interface for the application.
  - Include navigation elements or components to access different sections or features of the app.
  - Create forms for user input, such as registration, login, data entry, or search.
  - Display data to users in a structured and readable format.
  - Ensure your Web application works correctly in the latest HTML5-compatible browsers: Chrome, Firefox, Opera, Safari (latest versions, desktop and mobile versions).
- **Functionality**
  - Implement Web APIs or Semantic Web apps to access server-side data. These may include endpoints for listing, creating, updating, and deleting data.
  - Implement (React/Angular/etc.) components to manage the state of your application and its features.
  - Include at least two features that allows users to input data.
  - Include at least two features that displays, filters, or manipulates data.
  - Implement user interactions, such as button clicks, form submissions, or data updates.
  - Ensure the application functions as expected and is free of errors.
- **Data Management**
  - Design and implement a database schema or RDF store suitable for your project's needs.
  - Utilize data storage and retrieval mechanisms, such as state management (e.g., React Context, Redux) or APIs.
  - If applicable, persist data between sessions (e.g., using local storage).

Additional requirements

- Mandatory
  - Include application of at least **one Web tool** (wiki, blog, social network, etc.) in the project.
  - Use **GitHub** or other source control system as project collaboration platform. Each team member **should** contribute with commits.
- Recommended

    - Use any **project management tools** for collaboration e.g., Azure DevOps, Trello, Jira, Ambra, etc.
    - Use **communication platforms** for collaboration e.g., Discord, Slack, etc.
- Optional
    - Use **Progressive Web Apps** technology to provide single code multiplatform deployable code.
    - Include **other emerging technologies**.
    - You may **share your project** to get external feedback
        - Most shared and commented projects will get additional **bonus score**
    - Implement **advanced state management** with Redux or Context API.

Projects

You should **write your application from scratch**. However, you are also **encouraged** to use:

- Existing projects and frameworks like CMS systems, forum systems, photo album systems, etc.
- Single Page Apps frameworks like React, Angular, Vue, Ember, etc.
- JavaScript APIs for accessing the server-side data. **Use REST, JSON and AJAX calls**.

You may choose one of the projects below or a project by your choice approved by the course instructors.

Blog

Required functionalities:

- **View** all posts (optionally with paging).
- Adding **new posts** by the blog owner (after login or password protected). Each post must have **tags**.
- Adding **comments** for every post by visitors – each visitor must fill out his name, email (optionally) and comment text.
- Implement a sidebar holding a **list of posts** sorted by month / year / etc. and a list of the **most popular tags**.
- **Counter of visits** for each post.
- Functionality for **searching** by tags.

Admin functionalities:

- **User registration** and user profiles.
- **Admin panel**: add / edit / delete posts, comments, tags, etc.

Forum

Required functionalities:

- **View** all questions / categories (optionally with paging).
- Implement a simple **registration** for forum users.
- Adding **new question** by the forum users. Each question must have **tags** and **category**.
- Implement **categories** for the forum questions.
- Adding **answers** to the questions by the forum visitors – each visitor must fill out his name, email (optionally) and comment text.
- **Counter** for visits for each question.

Admin functionalities:

- **Admin panel**: add /edit /delete forum posts, tags, answers, categories.
- Functionality for **searching** by question, answer and tags.

- Implement **ranking** according to user activity.

Photo Album

Required functionalities:

- **View** all albums / categories (optionally with paging). **Browse** album photos.
- **Creating** new album in a category.
- **Uploading photos** (validating pictures size and type) / **downloading** photos.
- Adding **comments** to photos and albums.
- Implement album's **ranking system** (e.g. vote from 1 to 10 or like / dislike).
- Show the most **highly ranked** albums in a special section at the main page.

Admin functionalities:

- Implement **user registration**.
- Functionality for **searching** by album name / category.
- **Admin panel** (if registration is implemented): add / edit /delete albums, photos, comments.

Deliverables

Put the following in a **ZIP archive** and submit it (one team member submits the file):

- The complete **source code** of your project (JavaScript, jQuery, HTML, CSS, images, scripts and other files) with clear code comments.
- A **README file** with instructions on how to run the project locally.
- A **presentation** of your project (e.g. PowerPoint slides) of your project. It should provide the following information:
    - Project name and purpose – what you have created?
    - Team name, list of team members.
    - Contributions of each team member.
    - Technical description.
- **Github** link to the project (open for viewing)
- A **live demo** of the project (hosted on platforms like Netlify, Vercel, or GitHub Pages).
- **Logs** on any collaboration platform Discord, Trello, etc.

Public Project Defense

Each team will have to deliver a **public defense** of its work in front of the other students, trainers and assistants. Teams will have **only 10 minutes** for the following:

- **Demonstrate** the web application (very shortly).
- Show the **source code** and explain how it works.
- Explain how each team member has **contributed**: display the commit logs in the Source Control System you are using.
- Optionally you might prepare a **presentation** (3-4 slides).

Please be **strict in timing**! On the 10th minute you **will be interrupted** to start with a 10-minute question and answers session.

Be **well prepared** for presenting maximum of your work for minimum time. Bring your own laptop. Test it preliminary with the multimedia projector. Open the project assets and run the client app and APIs beforehand to save time.

Give Feedback about Your Teammates

You will be invited to **provide feedback** about all your teammates, their attitude to this project, their technical skills, their team working skills, their contribution to the project, etc. The feedback is important part of the project evaluation so **take it seriously** and be honest.