VueJS - Project Assignment

Your task is to design and implement a web application using VueJS.

Your application should make HTTP requests. You can use a service like Firebase or create your own Backend (there's no restriction in languages: Node.js/ASP.NET/Spring/Symfony/etc.). You can also use a free and open API service like Fake Store API, Pokemon API and many more (more examples here - Public APIs).

It can be a discussion forum, blog system, e-commerce site, online gaming site, social network, recipe app, or any other web application by your choice.

The application should have:

- public part (accessible without authentication)
- private part (available for registered users)

1. Application Structure

1.1 Public Part

The public part of your project should be visible without authentication. This public part could be the application start page, the user login and user registration forms, as well as the public data of the users, e.g. the blog posts in a blog system, the public offers in a bid system, the products in an e-commerce system, etc.

1.2 Private Part (User Area)

Registered users should have personal area in the web application accessible after successful login. This area could hold for example the user's profiles management functionality, the user's offers in a bid system, the user's posts in a blog system, the user's photos in a photo sharing system, the user's contacts in a social network, etc.

2. General Requirements

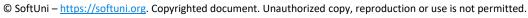
Your Web application should use the following technologies, frameworks and development techniques:

- 1. At least 3 different dynamic pages with routes (Vue Router). General pages like about, contacts, login, register do not count towards that figure. If your project doesn't cover this condition, you will not be graded!
- 2. Use VueJS for the client-side. You are not required to use only Composition API syntax, Options API is completely valid and accepted.
- 3. Communicate to a remote service (via REST, sockets, GraphQL, or a similar client-server technique).
- 4. Implement well known authentication method –Firebase/Backendless or with JSON Web Tokens (JWT's)
- 5. Use a source control system like GitHub, Bitbucket etc. Commit inside the repository for at least 3 days. Follow the best practices when committing to a repository (commit different features partially and create branches)

3. Other Requirements

- 6. Apply error handling and data validation to avoid crashes when invalid data is entered
- 7. Brief documentation on the project and project architecture (as Readme.md file)
- 8. Structure of the whole projects folders structure, naming of components, arrangement of services, layours, composables
- 9. Good usability (easy to use UI)



















4. Public Project Defense

Each student will have to deliver a public defense of their work in front of trainers. Students will have only 15 minutes for the following:

- **Demonstrate** how the application works (very shortly)
- Show the **source code** and explain how it works
- Answer of three theoretical questions

Please be strict in timing! On the 10th minute you will be interrupted, and you will be asked three short theoretical questions.

Be well prepared for presenting maximum of your work for minimum time. Open the project assets beforehand to save time.

5. Bonuses

- Use **Composition API** syntax
- Create and use **custom composable** functions
- Deploy the application in a **cloud environment** so it's accessible to users
- Use **Animations and Transitions** somewhere in your application
- Use state management for VueJS applications Pinia
- Anything that is not described in the assignment is a bonus if it has some practical use.

6. Assessment Criteria

General Requirements – 25 %

Functionality Presentation – 75 %

Adequately and clearly demonstrate the requested functionality. Know your way around the application and quickly demonstrate the code. Evaluation in this section is also based on the **structure** of your code.

- **Templates** Use data binding (one-way and two-way). Minimal amount of JavaScript expressions inside templates (use computed properties, watchers and methods instead). Use build-in directives (conditional rendering, list rendering, style & class bindings).
- Components Correct component registration (local or global). Pass data to child components with props, use an events to emit custom events between components. Use slots, dynamic, async components when needed and validate props with prop types or other equivalent techniques such as TypeScript.
- Forms Implement input bindings. Use input modifiers when needed. Implement front-end validation with **Vuelidate** or other equivalent.
- Routing Use Vue Router. Navigate with router links, setup child routes, redirects, named routes. Protect certain routes with Guards.
- **Project Architecture** Create a "service layer" for HTTP requests, order components and into subfolders and in a consistent matter.

Bonuses – Up to 10 %

Additional functionality or libraries outside the general requirements, with motivated usage.



















Additional functionality or libraries outside the general requirements, with motivated usage.

7. Submission Deadline

You must submit your project before 23:59 on 13 Dec using a survey that will show up on 07 Dec. A presentation schedule will be available on 14 Dec and will include only the projects that were submitted beforehand. Nonsubmitted projects will NOT be evaluated.

The Project Defense will be in Discord. You will receive link when the schedule is available.















