Vue.js Test Task

This task is meant to be short and practical. We're not evaluating layout or styling — super basic is totally fine.
The main goal is to see how you approach client-side app structure and SSR setup.

API server: https://modelsocietyapi-stage-a3dcfsb2hgf9fkd0.eastus-01.azurewebsites.net/

1. “Login” Page

- Create a simple email/password login form.
- No styling needed, just functionality.
- This page exists only so the rest of the task works — we need to be able to log in.
- Use Bearer authentication for protected endpoints.
- To log in, send a `POST` request to `/api/auth` with:

  ```
  {
    "email": "test566@gmal.com",
    "password": "1234qwer"
  }
  ```

- The response will contain `{ authToken }`.
- You'll need to implement an authentication mechanism to store and include this token in future requests.

2. “User Profile” Page

- Create a route like `/member/{nameForUrl}/profile`
- Use `/api/members/{nameForUrl}/profile` to fetch profile data. No auth required.
- Display:

    - User name
    - "About me" text
    - List of images (just render image URLs or basic `<img>`s — no styling needed)

- Add links on the page to two example profiles:

    - `/member/vova/profile`
    - `/member/Model-Society-Admin/profile`

Requirements:

- **SSR rendering** is required for first page load (for SEO).
- Navigating to other profile links should happen client-side (SPA-style, no full page reload).
- If the current user is authenticated, display their email at the top.
    - To get it, call `/api/me` (requires Bearer auth).
    - In the real app, this would return more user-specific data. So keep in mind that this is *dynamic*, non-SEO content.

✅ Your implementation should reflect that the profile page has both static SSR content (public profile info) and dynamic content (user-specific info like email).
✅ It's up to you how to design that interaction for good performance and SSR compatibility.
✅ Please explain your reasoning and approach.

3. Deployment

- Deploy the app somewhere with SSR support
- Explain why you picked that hosting.

4. Share Your Code

- Push the code to a public GitHub repository and share the link.