1 Requirements:

Node.js Installed on your computer see : https://nodejs.org/en/download
Install pnpm : npm install -g pnpm

2 Microfrontend Example : Create React App Example / Rsbuild

This example demos a basic host application loading remote component.

- ‘host‘ is the host application (cra-based).
- ‘remote‘ standalone application (cra-based) which exposes ‘Button‘ component.

3 Installing the application

- Run ‘pnpm install‘ to install all required files and node module. This will take a while.
- Run ‘pnpm install –save-dev jest‘ we will need it later to run tests on the app. (Jest will be our test container)
- Run ‘pnpm install cypress –save-dev‘ we will need it later to run tests on the app. (Cypress will run End to End Tests)

4 Building and Running Demo

Run ‘pnpm run start‘. This will build and serve both ‘host‘ and ‘remote‘ on ports 3001 and 3002 respectively. Also it will build all apps which respects the global package.json start scripts ”start”: ”pnpm –filter cra$_*start$”, $basicallyallappwhichstartsby'cra$

- localhost:3001 (HOST DASHBOARD ADMIN)
- localhost:3002 (STANDALONE REMOTE - DASHBOARD CLIENT)

5 Please always use pnpm and not yarn or npm

You will have to send me the code as a .zip file or a link to your github repository + the notion export as a .pdf

6 Next steps :

1. Create a Notion (or Confluence or Obsidian) page where you will write documentation just like a true feature teams. Ideally one (short) page with screenshots for each steps there:
    - Simply create a notion account and share your Notion's page with your project partner.
      https://www.notion.so/fr-fr
    - if you can add me as a contributor : idodgedit@gmail.com
2. Create a Shared Component App
    - Develop a shared library of React components.
    - Ensure these components are reusable and modular.
    - Create a Footer and a header in the shared component app. Use it in both host and remote Landing page (app.js)
    - Your files must be .JSX not .JS!
    - Ideally, expose it like Button/local button, else just import it in app.js
        - Copy remote project and call it ”sharedComponent”
        - In sharedComponent/package.json, rename the app to ”cra$_s$*haredcomponent*” *InsharedCompon*

– In cra/package.json, add the new folder under workspace

  - Remote App Development (Client's View)
      - Develop a landing page for Remote and Host. Using the footer and header from the SharedComponent App.
      - Create a unique component for remote and a unique one for Host. Ideally, one that is built with parameters (you can pass a title as a string)
      - Integrate Vitest for testing. Write basic tests for the component you created in remote and host.
      - check the folder StepsBySteps and use package.json
      - Create a new vitest folder inside the cra folder.
      - Use the package.json in this ”2-Create a vitest app folder”
      - follow guide : https://vitest.dev/guide/
      - remember you will need a new package.json for this folder and also a vitest config file.
      - Create two unit test, one for the component created in host, and one for the component created in remote. Don't test footer header from sharedComponents!
  - Host App Development (Client View)

    - Implement a fake API call to fetch and display data (use tools like JSONPlaceholder or create your own mock data). Use mockapi for a quick available api : https://mockapi.io/ or the pokemon api https://pokeapi.co/

- Add file upload (Admin View)
    - Use AWS S3 Api to upload files (any kind of files, pdf, images etc) to OVH public cloud.
    - Create your account and use the offer with 200€ free credits: https://www.ovhcloud.com/fr/lp/cloud-storage-trial-offer/
    - and then create a standard object storage https://www.ovhcloud.com/fr/public-cloud/object-storage/
    - you can also use aws S3 or scaleway instead of ovh.
- Read files from AWS S3 bucket (Client view)
    - See all uploaded files in a dashboard (or an array) fetched from OVH Public cloud

7 BONUS:

(a) Deploy the app on a free Platform as a Service
- You can use Vercel, fly.io, dokku, Qovery

(b) Make a dockerfile and run the app through docker.
- Deploy the app with a more complex paas which will use the dockerfile. (Dokku, Qovery etc.)