InsideSherpa Engineering Challenge

The goal of this task is to create a simple cloud-storage app for the web, conceptually similar to how Dropbox or Google Drive will work. You are required to write both the backend server and a web UI for this application. This task is expected to take around 2-3 hours to finish.

Although InsideSherpa is built on top of full-stack javascript, you are free to use either Java, Javascript (including Typescript/Flow), Python, Ruby, Go, or PHP. You are free to use any open source libraries/frameworks.

Backend server

You will need to implement a web-server capable of serving a simple page and accepting file uploads done on that page. Here are the requirements:

1. The web-server is accessible on `http://localhost:9393`
2. When a user visits said address on a web-browser, they will be served the web UI where they can upload files
3. Upon receiving the file, the backend server will store that file inside `<app-dir>/uploads` directory (`<app-dir>` is the root directory of the source code repository)
4. The file will be saved with the exact filename and file extension as uploaded by the user

Web UI requirements

You are required to design a single web-page with these requirements:

1. The UI displays a list of filenames already saved on the server, vertically and horizontally centered in the page
2. Uploads can be done **only** through a drag-and-drop interaction (user must drag a file from their desktop to the page to upload it, not through a button)
3. Upon successful upload, the web UI will update to show all the saved filenames including the new one

Extra points

If you have done all of the above, you can do these things for extra points:

1. A more professional looking UI
2. Better handling of duplicate filename uploads (big bonus point)
3. Turn the list of filenames in the UI into links that when clicked will download the file into your usual downloads folder (big bonus point)

Extra requirements and details

1. You **must** have a `<app-dir>/bin/start` bash script which we can use to run the web-application
2. You **may** have a `<app-dir>/bin/install` bash script which you can use to download necessary dependencies
3. You **may** include a compilation/build step in either the `<app-dir>/bin/start` or `<app-dir>/bin/install` script
4. You can assume that the computer that we will be using already has all this at latest version:
    - PHP and composer
    - Nodejs and npm
    - Python and pip
    - Java and maven
    - Ruby and rubygems
    - Go and dep
5. Testing will be done on a Mac using the latest version of Google Chrome (you can develop on any system, just make sure it runs on Macs as well)
6. Your solution must be committed to a git repository, you will then zip up the whole directory (which includes the .git folder) to be sent as submission

Marking criteria

1. Your code (including HTML and CSS) should be readable, maintainable, and easy to understand
2. Modular and testable code (even if your submission does not contain any tests) is preferred
3. Your source code directory structure should be clean and promote maintainability
4. Do write comments on places where it makes sense or can help us understand the solution better
5. Don't over-engineer the solution
6. Don't re-invent the wheel, do use open-source libraries if it fits the use-case

Most importantly

We do appreciate every effort you put into making the solution. A half-baked or hacky solution is much better than no solution at all. At the end of the day, we are a startup trying to build a great product while still delivering value to users at very fast pace. It is not unusual that trade-offs will have be made to save time and meet an important release cycle. Being able to know which compromise to take and which feature to prioritise during critical times is something that we really look for in our engineering culture.

Good luck and have fun :-)