Jupiter-SDK Assessment

(This assessment should be done within 5 days, starting from the day you receive it.)

Your goal is to create a minimal web application that allows text data typed into a textbox to be saved to a text file which will be stored in a file storage (e.g. local file system), and loaded back for viewing and atomic editing. The web application will then need to be run on a server for usage.

Requirements and assumptions

Work in a git repository so that your code is version controlled.

Web Application Server

- Although using Node.js is encouraged, feel free to use any language, framework, and libraries you find useful. Keep it simple so you can finish quickly.
- The application does not need to be “production-ready” or secured, and can operate in a debug mode if preferable.
- All users are anonymous and no user authentication is necessary.
- Any simple database can be used to save data to. Do not be concerned with scaling issues. For example, using SQLite is fine.
- No tests or documentation is necessary.
- Run the webserver on port 8080.

Web Application pages

There are at least three HTML pages, as detailed below. CSS styling is unnecessary. Simple HTML is fine.

<u>“New” page</u>

- This is the default page of the webserver when it is accessed.
- It should have a form containing three input fields:
    - A text input, to type file name into. (forget about the potential conflict, let's assume the file name will always be unique)
    - A textarea, to type text into.
    - A “Save” button to submit the form.

- Once the form is submitted, your application server should save the text data as a text file (in the local file system or any cloud storage service).
- After saving the text file, create a database entry with all the data needed to edit the file in the future (see “Edit” page below). Also assign a unique identifier to this file in your database, such as a number.

“View” page

- This page has a unique URL to show all the text files previously submitted.
- Clicking the file name should trigger browser to download the file.
- Each file has a corresponding "Edit" button by its side.

“Edit” page

- This page has a unique URL to edit a text file previously submitted.
- This page should have a form containing a textarea input field with the contents of the file.
- Any user accessing this page causes the web application server to download file content dynamically, to include in the textarea field.
- A “Save” button on the web page should submit theform.
- Once the form is submitted, your application server should update the text file accordingly.

Locking

- To prevent conflicts when multiple users edit the same file through your web application, any user editing the file is given a one minute lease to do so, during which time no other user can edit the file.
- When a user loads the edit page for a specific file, associate the text file to that editor for the next 60 seconds to “lock” it.

The application you submitted should be runnable (whether is from a local dev server or a cloud platform, such as heroku), and you need to provide a detailed description of how to run it.