PassFort
A Moody's Analytics Company

Wiki Coding Task

We'd like you to create the frontend for a wiki, like Wikipedia. A description of the wiki follows:

1. A wiki is a collection of **documents**
2. **Documents** are lumps of plain text. No graphics, attachments, or formatting
3. Each **document** is uniquely identified by a **title** that is a maximum of 50 characters in length. This **title** does not change for the life of a document
4. A **document** can have multiple **revisions**, as it is updated over time. We store all historical **revisions** of a document
5. We should be able to view the **document** as *it was* at any point in time. I.e. we can use any timestamp to fetch a **revision** e.g. If we have a document at time 1pm and time 3pm, then sending a timestamp of 2pm should return the document as it was at time 1pm.

Your task is to implement a Single Page Application (SPA) with the following features:

1. Show a list of documents
2. Allow you to click on a document and see:
   a. The contents of the latest revision (formatted with markdown)
   b. A list of the supplied revisions
1. Allow you to choose a revision, and view the document at that time
2. Allow you to post a new revision of the document

Technical implementation requirements:

- We ask that you spend <u>**only 2.5 hours**</u> on this task. Please start the task by initialising a git repository locally. At the end of this timeframe create a git bundle by running git bundle create passfort.bundle master and send this to us (this is to ensure all candidates are evaluated fairly).
- The code should be production ready
  - It should handle API errors, and failures
- You should write some tests around your application
  - It is up to you to decide which tests and how to write them
- You should include routing (e.g. I can refresh the page, and it will refetch the document and revision I was viewing)

You have been supplied with a python project, which implements a backend for the above. It surfaces a JSON API. It has CORS enabled, and can be accessed at localhost:5003. You can run it using run.sh; it requires python. It is intentionally undocumented, all endpoints are defined within the code.

You can use any technologies you like. Our current stack is using react with some redux and react-refetch, but you are welcome to choose anything you like, even compile to JS languages like: Elm, Purescript, or Reason. We recommend you use a starter kit to get things like bundling sorted for you; but please put a link to it in a README file in your task.

How you'll be assessed:

We're not expecting complete implementation, and the degree of completion will vary around experience level. We're more interested in you showcasing best practices and attention to detail, rather than completing the whole task poorly. Treat it like you would any production code.